

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	Кафедра
вычислительной техники	МОиПЭВМ
Направление подготовки	09.03.02 «Информационные системы и технологии»
Направленность	Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА
на тему
Программный сервис для неформального взаимодействия
участников образовательного процесса ВУЗа

Студент	_____	<u>Егорова Валентина Григорьевна</u>
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	<u>Шибанов С.В.</u>
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролёр	_____	<u>Такташкин Д.В.</u>
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	<u>Козлов А.Ю.</u>
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	<u>Попова Н.А.</u>
	(подпись)	(фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов «25»
декабря 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Егорова Валентина Григорьевна гр. 20ВИ1 факультета ФВТ

2. Тема работы Программный сервис для неформального взаимодействия участников образовательного процесса ВУЗа

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Шибанов С.В.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение: программный сервис для неформального взаимодействия участников образовательного процесса ВУЗа предназначен для улучшения коммуникации между преподавателями и студентами и повышения качества образования за счёт развития неофициальных отношений между обучающимися и педагогами.

Среда выполнения: операционная система Microsoft Windows 10

Средства разработки: IntelliJ IDEA Community Edition 2024, Visual Paradigm Enterprise 2015, ArgoUML

Язык программирования: Java, SQL

Технологии разработки: UML, ООП, HTML, Spring

4. Перечень подлежащих разработке вопросов _____

Анализ предметной области программного сервиса и постановка задачи на разработку _____

Проектирование сервиса для неформального взаимодействия участников образовательного процесса ВУЗа _____

Обоснование выбора средств разработки сервиса _____

Разработка базы данных и клиент-серверной части сервиса _____

Тестирование программных средств программного сервиса _____

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2024 – 13.04.2024	
2	Проектирование программных средств	20%	15.04.2024 – 25.04.2024	
3	Реализация программных средств	30%	26.04.2024 – 12.05.2024	
4	Тестирование программных средств	20%	13.05.2024 – 22.05.2024	
5	Оформление пояснительной записки	20%	23.05.2024 – 03.06.2024	

Дата выдачи задания «21» декабря 2023г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 146 листов, 30 рисунков, 13 таблиц, 23 использованных источника и 5 приложений.

СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, СЕРВИС, НЕФОРМАЛЬНОЕ
ВЗАИМОДЕЙСТВИЕ, ОБРАЗОВАТЕЛЬНЫЙ ПРОЦЕСС, ВУЗ, JAVA, SQL, UML,
БАЗА ДАННЫХ.

Объект исследования: программный сервис для неформального взаимодействия участников образовательного процесса ВУЗа.

Цель работы: разработать сервис, предназначенный для реализации неформальных отношений участников образовательного процесса ВУЗа.

Технологии разработки: IntelliJ IDEA, Java, UML, Visual Paradigm, H2, Spring.

Задание: проанализировать предметную область, спроектировать программные средства, разработать базу данных и клиент-серверную части сервиса для неформального взаимодействия участников образовательного процесса ВУЗа.

Результаты работы: был спроектирован и реализован сервис для неформального взаимодействия участников образовательного процесса ВУЗа.

					ПГУ 09.03.02 – 8БР201.07 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программный сервис для неформального взаимодействия участников образовательного процесса ВУЗа Пояснительная записка	Лит.	Лист	Листов
Разраб.		Егорова В.Г.						
Провер.		Шибанов С.В.					4	146
Реценз.						группа 20ВИ1		
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение	7
1 Анализ предметной области и постановка задачи на разработку программного сервиса для неформального взаимодействия участников образовательного процесса ВУЗа.....	9
1.1 Взаимодействие участников образовательного процесса ВУЗа	9
1.1.1 Категории участников образовательного процесса ВУЗа	9
1.1.2 Виды взаимодействия участников образовательного процесса ВУЗа	11
1.1.3 Особенности неформального взаимодействия участников образовательного процесса ВУЗа	12
1.2 Сравнительный анализ приложений-аналогов	13
1.2.1 Цели и задачи сравнительного анализа	13
1.2.2 Выбор приложений-аналогов для анализа	14
1.2.3 Обзор приложений-аналогов.....	16
1.2.4 Результаты сравнительного анализа	21
1.3 Постановка задачи на разработку сервиса	23
1.3.1 Цели разработки и назначение сервиса	23
1.3.2 Классы и роли пользователей сервиса.....	24
1.3.3 Автоматизируемые бизнес-процессы	25
1.4 Планирование процесса разработки сервиса	28
2 Проектирование сервиса для неформального взаимодействия участников образовательного процесса ВУЗа	32
2.1 Моделирование функциональных возможностей сервиса	32
2.2 Разработка концептуальной модели сервиса	40
2.3 Разработка программной архитектуры сервиса	42
2.4 Разработка прототипа пользовательского интерфейса.....	45
2.5 Разработка требований к аппаратно-программному окружению	47
2.6 Обоснование выбора средств разработки	48
2.6.1 Обоснование выбора средств автоматизации	48
2.6.2 Обоснование выбора инструментальных средств разработки программного обеспечения	50

2.6.3	Обоснование выбора языковых средств разработки	51
2.6.4	Обоснование выбора средств управления данными	54
3	Разработка программного обеспечения сервиса для неформального взаимодействия участников образовательного процесса ВУЗа	56
3.1	Разработка базы данных	56
3.2	Разработка серверного программного обеспечения	60
3.3	Разработка клиентского программного обеспечения	62
4	Контроль качества программного обеспечения сервиса для неформального взаимодействия участников образовательного процесса ВУЗа	66
4.1	Планирование процесса контроля качества программного обеспечения	66
4.1.1	Обоснование выбора модели тестирования	66
4.1.2	Обоснование выбора метрик программного кода	68
4.2	Тестирование программных средств	70
4.2.1	Тестирование белым ящиком	70
4.3	Оценка качества программного кода на основе метрик	72
4.3.1	Метрика SLOC	72
4.3.2	Метрика Холстеда	73
4.3.3	Метрика Чепина	75
	Заключение	78
	Список использованных источников	79
	Приложение А (справочное) Глоссарий	82
	Приложение Б (рекомендуемое) Шаблоны пользовательского интерфейса	85
	Приложение В (рекомендуемое) Код разработки базы данных	90
	Приложение Г (рекомендуемое) Листинг серверной части сервиса	96
	Приложение Д (рекомендуемое) Листинг клиентской части сервиса	129

Введение

Повышение качества обучения и создание благоприятной для получения образования среды, всегда является актуальным. При этом очень важно, чтобы улучшения происходили не только со стороны формальных отношений между участниками образовательного процесса, но и неформальных. В современном мире процесс обучения включает в себя не только передачу знаний, но и взаимодействие между студентами и преподавателями, обмен опытом и идеями, а также участие в общественной жизни университета.

Из-за выбора, в основном, неэффективных методов связи студентов с преподавателями, наличия затруднённости в поиске источников помощи у студентов, особенно в первое время обучения в ВУЗе, качество обучения не может в полной мере достичь высоких результатов. Для решения данной проблемы был разработан программный сервис для неформального взаимодействия участников образовательного процесса ВУЗа.

Целью данной квалификационной работы является создание сервиса, который сможет в большей мере учесть все особенности неформальных отношений преподавателей и студентов и обеспечить их эффективное взаимодействие.

Для выполнения данной цели были поставлены и решены следующие задачи:

- проанализирована предметная область и поставлены задачи на разработку сервиса;
- выполнено планирование процесса разработки сервиса;
- выполнение проектирование программного сервиса;
- разработано программное обеспечение (база данных, клиент-серверная часть приложения);
- протестированы внутренняя структура и пользовательский интерфейс программного сервиса.

В результате выполнения дипломной работы был разработан программный сервис для неформального взаимодействия участников образовательного процесса ВУЗа, который позволяет:

- неформально общаться со студентами и преподавателями;
- размещать заявки о проблемах, связанных с учебными вопросами и оперативно получать на них ответ;
- организовать библиотеку личных файлов и найти необходимые материалы по удобным для участников образовательного процесса критериям как в своём архиве, так и в документах других пользователей.

Разработка программного сервиса происходила на языке Java с помощью IDE IntelliJ IDEA Community Edition 2024.1.2.

1 Анализ предметной области и постановка задачи на разработку программного сервиса для неформального взаимодействия участников образовательного процесса ВУЗа

1.1 Взаимодействие участников образовательного процесса ВУЗа

1.1.1 Категории участников образовательного процесса ВУЗа

Образовательный процесс высшего учебного заведения представляет собой воспитательную деятельность организации, главной целью которой является обучение студентов всеми знаниями и навыками, необходимыми для эффективного выполнения определённых профессиональных обязанностей и содействие обучающемуся в приобретении умения применять эти знания на практике [1].

Важно отметить, что учебная организация направлена на формирование у обучающихся мотивации обучаться и обновлять свои знания в течение всей жизни, что требует создания привлекательной и интересной для студента образовательной среды.

Для успешного хода выполнения целей учебного процесса ВУЗа требуется эффективное взаимодействие всех участников, играющих ключевую роль в формировании образовательной среды. В общеобразовательном учреждении ими являются обучающиеся, педагогические работники образовательной организации, родители (законные представители) обучающихся. У каждой категории лиц, принимающих участие в образовательном процессе, есть свои характерные черты, права и обязанности.

Студент – лицо, в установленном порядке зачисленное в высшее учебное заведение для обучения. Является активным участником учебного процесса ВУЗа [2]. Студенты имеют право на получение знаний, соответствующих современному уровню развития науки, техники и культуры; бесплатно пользоваться библиотеками, информационными фондами, услугами различных подразделений университета; участвовать в разнообразных видах научно-исследовательской деятельности т. д.

В обязанности студентов входит усвоение знаний, выполнение всех заданий, предусмотренных учебным планом, соблюдение устава ВУЗа, правила внутреннего распорядка и общежития.

Педагогический работник – лицо, профессионально занимающееся образовательной деятельностью. В состав педагогических работников входят преподаватели, методисты, административно-педагогический персонал (деканаты, кафедры и т. д.) и другие участники [3]. В права и свободы работников высших учебных заведений входит свобода преподавания, право на разработку собственных методик обучения, право на участие в разработке образовательных программ и учебных планов и другие права, предусмотренные законодательством РФ.

Педагогические работники обязаны соблюдать устав образовательной образовательного учреждения, обладать профессиональными навыками и систематически повышать квалификацию, качественно реализовывать рабочую программу учебных дисциплин. А также обязанностью является развитие у студентов познавательной активности, инициативы и формирование ответственного отношения к профессиональной деятельности.

Родители (законные представители) несовершеннолетних обучающихся имеют право, учитывая мнение ребёнка, выбирать форму и место получения образования; ознакомиться с уставом и учебной программой университета; принимать участие в управлении образовательной организацией.

Родители (законные представители) несовершеннолетних обязаны обеспечить получение детьми общего образования, соблюдать правила, установленные учебной организацией.

Все вышеперечисленные категории активно взаимодействуют между собой, формируя работу образовательного процесса и оказывая влияние на качество образования и успешное обучение студентов. В рамках дипломной работы будут рассматриваться категории студентов и преподавателей, а также их взаимоотношения.

1.1.2 Виды взаимодействия участников образовательного процесса ВУЗа

Для поддержания полноценной работы учебного учреждения и обеспечения всех необходимых результатов образовательного процесса участники постоянно взаимодействуют между собой.

В соответствии с тем, каким образом категории лиц сотрудничают друг с другом, по своим характеристикам, целям и способам осуществления отношения делятся на следующие виды:

- формальное взаимодействие;
- неформальное взаимодействие.

Формальные связи устанавливаются на основе законодательства РФ и регламентируются различными уставами, положениями, инструкциями и другими нормативно-правовыми актами [4]. В случае образовательного учреждения официальное взаимодействие регулируется правилами, структурами и уставом, утверждёнными данным университетом.

Цель формального взаимодействия – обеспечение качественной и стабильной организации учебного процесса и решение основных задач образовательного учреждения. Такие задачи могут включать в себя выполнение учебных заданий, оценку их качества и соблюдения сроков сдачи, проведение занятий.

Примерами данного вида взаимодействия являются лекции, практические занятия, семинары, сдачи лабораторных работ, задолженностей, консультации по предметам, зачетные и экзаменационные сессии.

Таким образом, формальный вид взаимодействия происходит только в рамках установленных положений образовательного процесса ВУЗа.

Неформальное взаимодействие основано на неофициальных отношениях участников образовательного процесса ВУЗа. Данные отношения строятся на основе личных интересов и предпочтениях. Чаще всего неформальная форма общения возникает в дополнение формальному взаимодействию.

1.1.3 Особенности неформального взаимодействия участников образовательного процесса ВУЗа

Неформальное взаимодействие не имеет законодательной основы, не регулируется правовыми положениями и основано на личных интересах и предпочтениях. В рамках образовательного процесса ВУЗа неофициальные отношения между участниками представляют собой сотрудничество по вопросам, не регламентированным в организации. Общение происходит вне рамок учебного процесса.

Цели неформального взаимодействия могут быть самыми разнообразными: потребность в дополнительной поддержке и помощи; обмен опытом, идеями, знаниями; создание доверительных отношений; получение мотивации.

Исходя из результатов выполнения поставленной цели неофициальные отношения могут проявляться в различных формах:

- общение и дружеские отношения;
- совместные мероприятия;
- консультации и поддержка.

Вне учебного процесса участники могут общаться на различные темы, связанные с личными интересами и увлечениями.

Студенты, преподаватели и другие лица образовательной организации могут участвовать в научных конференциях, студенческих объединениях, спортивно-оздоровительных мероприятиях и т. д. Внеучебная совместная деятельность позволяет расширить круг увлечений и полезного досуга, сближая между собой всех её участников.

Студенты могут обращаться за помощью к своим преподавателям или задавать дополнительные вопросы для изучения проблемы, связанной с учебным процессом. Преподаватели, в свою очередь, предоставят дополнительные материалы или, опираясь на свои углубленные знания и опыт, могут подсказать, как решить данный вопрос.

Таким образом, неформальное взаимодействие позволяет решать такие задачи образовательного процесса, какие сложно разрешить при формальном взаимодействии. Оно способствует более эффективному и успешному обучению, раскрытию потенциала каждого участника учебного процесса. Поэтому неформальное взаимодействие также играет важную роль в создании благоприятной образовательной среды, как и официальные отношения.

1.2 Сравнительный анализ приложений-аналогов

1.2.1 Цели и задачи сравнительного анализа

Цель сравнительного анализа приложений-аналогов – выявление характерных различий между сервисами, предоставляющими неформальное взаимодействие между участниками образовательного процесса ВУЗа. Это необходимо для разработки наиболее актуального для пользователей и высококачественного программного продукта.

Для достижения данной цели существует определённый круг задач, которые нужно решить:

- 1) составление списка критериев, необходимых для сравнения приложений, направленных на наиболее полный анализ в соответствии с характерными чертами сервисов;
- 2) поиск программных продуктов, подходящих для сравнительного анализа;
- 3) обзор приложений-аналогов, краткое описание их характеристик по утверждённым критериям;
- 4) получение итоговых результатов сравнительного анализа информационных систем в наглядном представлении.

Для того, чтобы более целенаправленно провести обзор и изучить характерные качества приложений, требуется сравнивать по критериям, подходящим под данный анализ:

1) удобство использования (оценка интерфейса программного продукта, интуитивности дизайна, легкой навигации по системе, простота в использовании для различных категорий пользователей);

2) функциональность (чаты, форумы, обмен файлами, создание групповых обсуждений, личный профиль);

3) связь участников образовательного процесса (возможность и удобство общения напрямую студента с преподавателем и между студентами, отправление звонков);

4) организация помощи и поддержки (распространение личных документов в открытом доступе для всех пользователей, создание и решение проблемы или вопроса, касающегося учебной деятельности);

5) организация структуры учебного хранилища (загрузка файлов, их обмен, хранение, поиск по критериям: группа, курс, год, семестр).

Выполнив сравнительный анализ систем по выбранным критериям, можно наиболее качественно решить задачи и достичь поставленной цели.

1.2.2 Выбор приложений-аналогов для анализа

В ходе исследования приложений-аналогов был сделан вывод о том, что все системы, каким-то образом касающиеся неформальных отношений образовательного процесса ВУЗа, по своим характеристикам можно поделить на несколько видов:

- социальные сети;
- электронные информационно-образовательные среды (ЭИОС) университета;
- интернет-платформы для дополнительной помощи при обучении.

Социальная сеть – это приложение, предназначенное для неформального общения пользователей вне зависимости от статуса или роли, обмена разного рода информацией, создания отношений между людьми, имеющими схожие интересы. К этому виду систем относятся Вконтакте, WhatsApp, Одноклассники и другие соцсети.

Электронная информационно-образовательная среда (по др. ЭИОС) – система, которая объединяет различные информационные ресурсы и инструменты для удобного взаимодействия студентов, преподавателей и сотрудников университета. К ней относятся ЭИОС ФГБОУ ВО «Пензенский государственный университет», ЭИОС ФГБОУ ВО «Московский государственный университет им. Н.П. Огарёва» и другие.

Интернет-платформа для дополнительной помощи при обучении – это информационная система, предоставляющая дополнительные возможности в образовательном процессе. Данные сервисы предоставляют спектр учебных ресурсов и материалов, не содержащихся в основной программе обучения. К такому виду информации могут относиться образовательные курсы, видеоуроки, тесты, вебинары, а также средства для общения с преподавателями или другими студентами. Примером интернет-платформы выступают образовательные площадки «Skillbox», «Campus»; онлайн-школы «Фоксфорд», «Skyeng»; пространство для учёбы и общения «Сферум» [5].

В качестве систем, предназначенных для сравнительного анализа, были выбраны следующие бесплатные веб-приложения: ЭИОС ПГУ, Вконтакте и Сферум.

Общие сведения о выбранных системах представлены в таблице 1.

Таблица 1 – Общие сведения приложений

<i>Приложение/ Характеристика</i>	ЭИОС ПГУ	Вконтакте	Сферум
Краткое описание	Информационная система Пензенского государственного университета, обеспечивающее обучение студентов в электронном виде	Популярная социальная сеть для общения, поиска друзей и обмена новостями	Закрытое и безопасное пространство для учёбы и общения педагогов, учеников и их родителей
Владелец	ПГУ	Mail.ru Group	ООО «Цифровое образование»
Год создания	2015	2006	2021

Продолжение таблицы 1

<i>Приложение/ Характеристика</i>	ЭИОС ПГУ	Вконтакте	Сферум
Вид ПО	Web-приложение, Мобильное приложение	Web-приложение, Мобильное приложение, Desktop-приложение	Web-приложение, Мобильное приложение

В следующем пункте представлен более подробный обзор приложений-аналогов с краткой информацией по каждой системе.

1.2.3 Обзор приложений-аналогов

В процессе изучения аналогов сервиса неформального общения участников образовательного процесса ВУЗа для сравнительного анализа были выбраны следующие приложения: социальная сеть «Вконтакте», электронная информационно-образовательная среда «ЭИОС ПГУ», образовательная платформа «Сферум». Ниже представлен обзор систем в соответствии с критериями, описанными в пункте 1.2.1.

Вконтакте – это крупная российская социальная сеть, основанная в 2006 году. Доступен на 82 языках; пользуется популярностью, в основном, у русскоязычных пользователей. На рисунке 1 продемонстрирована страница «Новости» данного сервиса, которая открывается при авторизации.

Социальная сеть «Вконтакте» имеет интуитивно понятный интерфейс, дизайн сайта лаконичный, оформление не перегружено. Несложная навигация по системе (между сообществами, личными страницами и другими элементами).

В приложении доступна возможность отправки сообщений (чаты) как отдельному собеседнику, так и нескольким; создание групп, звонков; организация личной информации в виде профиля и её редактирование; хранение фото-, видео-, аудиофайлов и их обмен; создание, редактирование, удаление постов и другое.

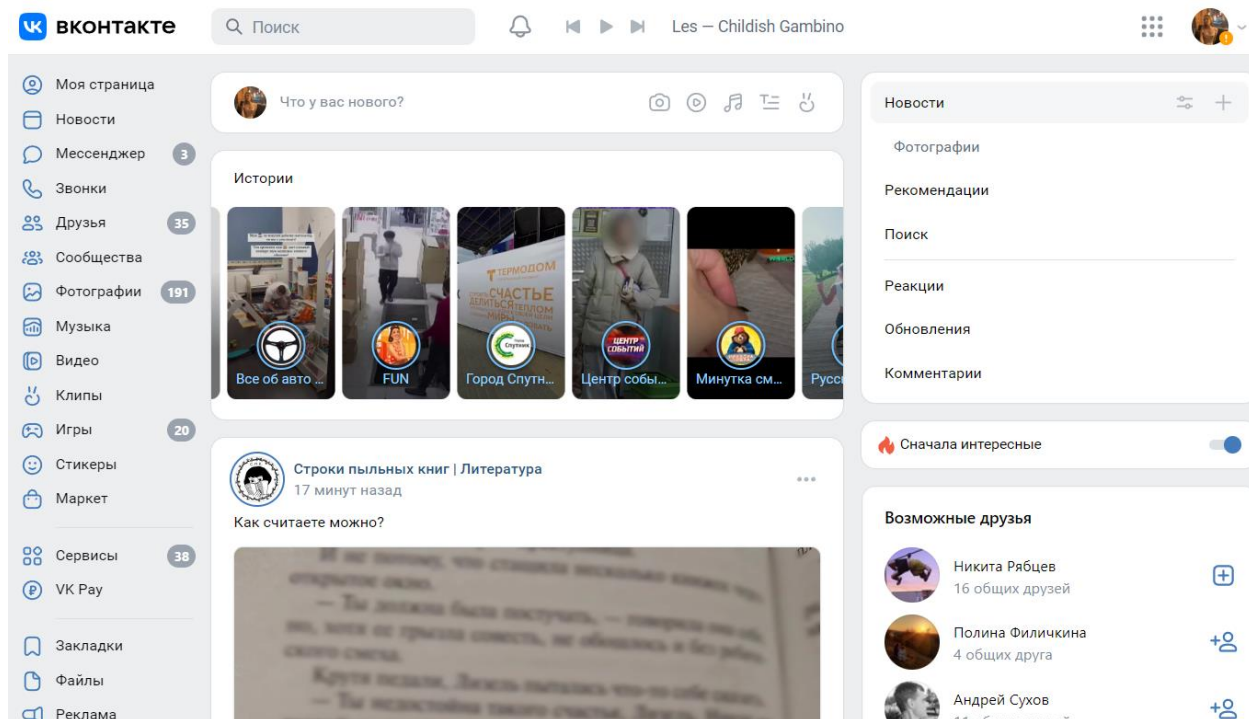


Рисунок 1 – Страница «Новости» социальной сети «Вконтакте»

Студенты могут общаться напрямую с преподавателями через личные сообщения в мессенджере, а также комментарии к постам. Возможны индивидуальные и групповые звонки при использовании функции аудио- и видеосвязи.

Преподаватели и студенты могут делиться своими материалами, сохранив у себя на личной странице или в группах. Создание и решение вопросов, касающихся учёбы, могут проходить в личных и групповых чатах, комментариях под постами или обсуждениях сообществ.

Загрузка и обмен файлами возможны через их публикацию на личной странице, в сообществе, также их можно прикрепить в сообщении собеседнику. Для удобной работы и структуризации документов есть метки, по которым файл можно будет найти при помощи поиска; есть возможность выбора доступа к файлу (личный, учебный, книга, другой).

ЭИОС ПГУ – электронная образовательно-информационная среда, содержащая в себе комплекс образовательных материалов и технологий и обеспечивающая доступ к учебным программам и курсам. На рисунке 2

представлен профиль пользователя ЭИОС, на рисунке 3 – кабинет Moodle, появляющийся при открытии вкладки «Учебный процесс».

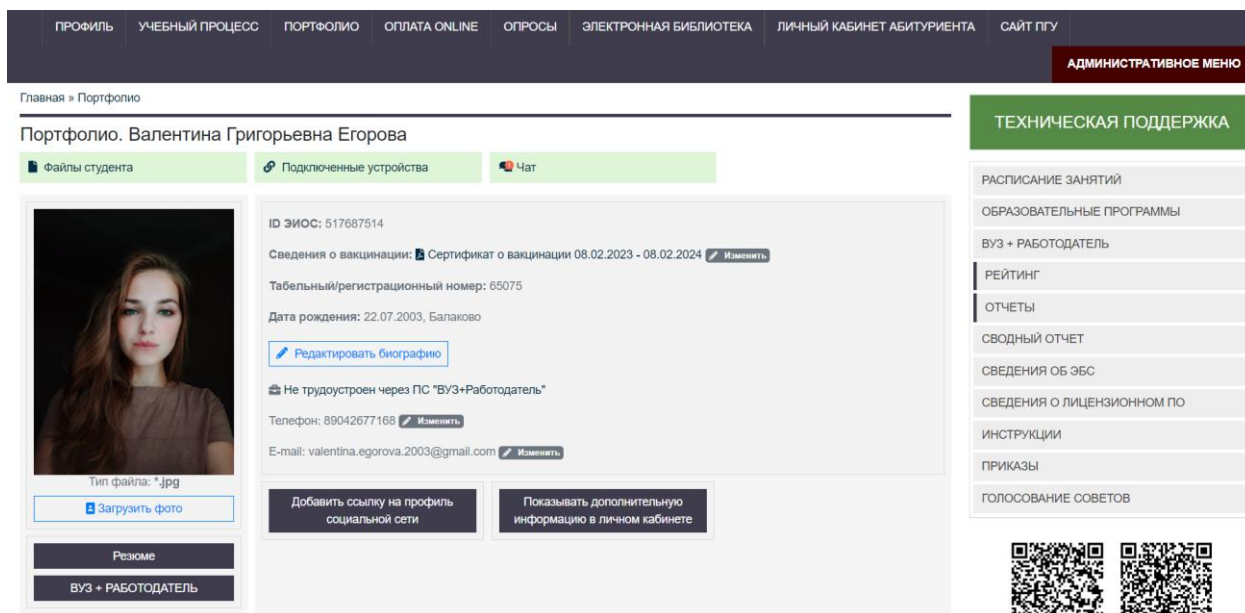


Рисунок 2 – Профиль пользователя ЭИОС

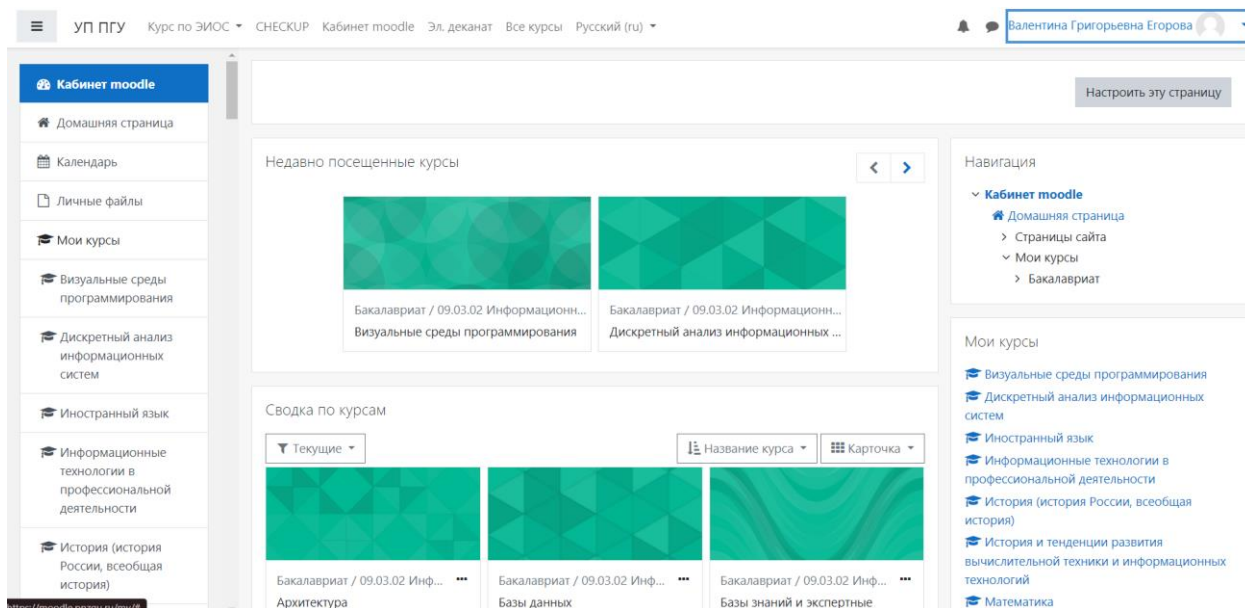


Рисунок 3 – Кабинет Moodle

ЭИОС Пензенского государственного университета имеет интуитивно понятный интерфейс; дизайн сайта лаконичный, оформление слегка перегружено. Навигация по системе немного затруднена, не всегда можно сразу найти необходимый раздел или материал.

В ЭИОС многие функции ориентированы под взаимодействие студента с преподавателем. Личная информация организована в виде профиля с возможностью редактирования. Доступен просмотр своих предметов, оценок. Отправка сообщений только преподавателю лично или через обсуждения в разделе с конкретным предметом. Также к функционалу системы относятся: хранение фото-, видео-, аудиофайлов и их обмен; создание, редактирование, удаление обсуждений и другое.

Студенты могут общаться между собой и напрямую с преподавателями через личные сообщения в учебном процессе ЭИОС или через создание обсуждения, входящего в конкретный курс.

Преподаватели и студенты могут делиться своими материалами, сохранив у себя в разделе «Личные файлы». Также преподаватели могут распространять конкретные методические документы, аннотации, рабочие программы дисциплины и другие файлы на странице предмета, находящийся под его ведением; создание и решение вопросов, касающихся учёбы, могут проходить в личных чатах или обсуждениях дисциплины.

Загрузка и обмен файлами возможны через их публикацию в разделе «Личные файлы», также их можно прикрепить в сообщении собеседнику. Для удобной работы и структуризации документов есть возможность создания папок, указание автора и выбор лицензии («Лицензия не указана», «Все права защищены», «Общественное достояние» и др.).

«Сферум» — это информационно-коммуникационная платформа, разработанное для дополнительной поддержки в обучении. Данное приложение было создано Минпросвещения и Минцифры в соответствии с постановлением Правительства РФ в целях реализации нацпроекта «Образование». И предназначено оно, в основном, для взаимодействия учеников с учителями. На рисунке 4 отображена главная страница пространства «Сферум».

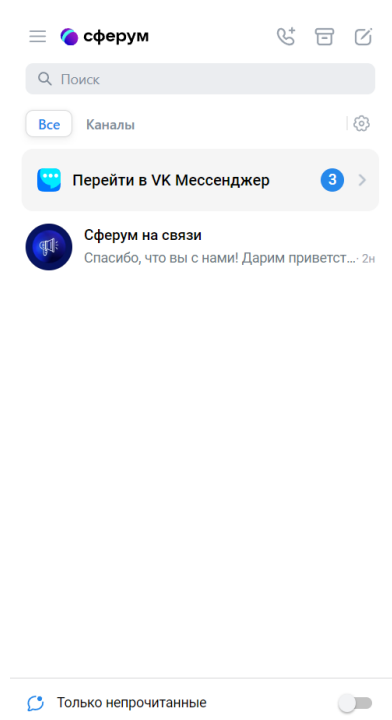


Рисунок 4 – Главная страница «Сферум»

Пространство «Сферум» имеет интуитивно понятный интерфейс. Дизайн сайта достаточно простой, оформление не перегружено. Есть удобное непереполненное меню.

В приложении многие функции ориентированы под взаимодействие ученика с учителем. Личная информация организована в виде профиля с возможностью редактирования. Доступен просмотр своих предметов, оценок через электронный журнал. Отправка сообщений возможна только педагогу лично или через групповой чат. В функционал платформы ещё входят создание, редактирование, удаление чатов и другое.

Ученики могут общаться между собой и напрямую с учителями через личные сообщения или в групповых чатах.

Ученики и педагоги могут делиться своими материалами в чатах. В «Сферум» разработана функция «Интерактивная доска», с помощью которой учитель может наглядно объяснить какой-либо вопрос или решить проблему.

Загрузка и обмен файлами возможны через их прикрепление в сообщении собеседнику. Для удобной работы с другими участниками есть возможность создания папок с чатами. В «Сферум» разработана функция «Сбор файлов», с

помощью которой можно собирать и хранить документы в отдельном месте и конкретных участников.

1.2.4 Результаты сравнительного анализа

На основе обзора приложений-аналогов была создана итоговая таблица 2, в которой отображены результаты сравнительного анализа по заданным критериям.

Таблица 2 – Итоговые результаты сравнительного анализа

<i>Приложение/ Критерий</i>	ЭИОС ПГУ	Вконтакте	Сферум
Удобство использования	Навигация по системе немного затруднена и перегружено меню.	Слегка перегружено меню.	Удобный интерфейс.
Функциональность	Есть много функций, ориентированных на учебную и рабочую деятельность студента и преподавателя.	Для всех пользователей один функционал.	В наборе только базовые функции.
Связь участников образовательного процесса	Преподаватели и студенты могут взаимодействовать через личные сообщения учебного процесса и обсуждения под конкретным предметом. Между собой студенты могут общаться только через личный чат.	Преподаватели и студенты могут взаимодействовать через личные сообщения, групповые чаты. Есть возможность создания аудио, видео звонков. Также организация связи может происходить через сообщества и обсуждения в них.	Преподаватели и студенты могут взаимодействовать через личные сообщения, групповые чаты. Есть возможность создания аудио, видео звонков.
Организация помощи и поддержки	Помощь оказывается в личных чатах с прикреплением материалов. Возможно решение проблемы в конкретном обсуждении дисциплины.	Помощь и поддержка по учебным вопросам оказывается в личных и групповых чатах, комментариях под постами или обсуждениях сообществ.	Помощь и поддержка по учебным вопросам оказывается в личных и групповых чатах. Есть функция «Интерактивная доска».

Продолжение таблицы 2

<i>Приложение/ Критерий</i>	ЭИОС ПГУ	Вконтакте	Сферум
Организация структуры учебного хранилища	Указание автора, и выбор лицензии при организации файлового хранилища. Есть раздел «Личные файлы».	Создание меток и выбор уровня доступа к ресурсу при организации файлового хранилища. Есть разделы «Файлы» и «Закладки».	Организация чатов по папкам. Есть функция «Сбор файлов».

В ходе анализа был сделан вывод о том, что базовые функции присутствуют во всех трёх приложениях, участвующих в сравнительном анализе, а именно: личный кабинет, функции по работе с сообщениями, операции с файлами и т. д.

В качестве функционала, необходимого для разработки сервиса, в системах были выделены следующие функции:

- в приложениях ЭИОС ПГУ и «Вконтакте» есть отдельный раздел, предназначенный для хранения и структуризации файлов;
- в «Вконтакте» разработаны метки и уровни доступа для быстрого нахождения необходимого документа;
- у «Сферума» представлена интересная функция «Интерактивная доска» для более наглядного решения вопросов;
- у «Вконтакте» и «Сферум» преимущество перед электронной образовательной средой в организации неформального взаимодействия в виде аудио и видео звонков.

В дальнейшем при разработке сервиса планируется ввести функцию «Хелп-доска», в которой будет организована помощь студентам и преподавателям по учебным и неучебным вопросам в виде тредов. Тред – это ветка обсуждений под сообщением, где пользователи общаются на конкретную тему [6]. Доски с проблемами будут разделены по темам «Мои», «Помогаю». У каждого пользователя будет установлен статус престижа, который будет формироваться от количества и качества помощи, оказанной участникам. Ещё

одной новой функцией будет организация более быстрого поиска учебных материалов. Она представлена в виде фильтрации по курсу, семестру, предмету, преподавателю.

1.3 Постановка задачи на разработку сервиса

1.3.1 Цели разработки и назначение сервиса

Разрабатываемый сервис предназначен организации свободного и неформального общения, обмена информацией, совместной работы между студентами, преподавателями и другими участниками образовательного процесса. Сервис позволит улучшить коммуникацию студента и преподавателя, повысить эффективность обучения и усвоение учебной программы. Он полностью учитывает все особенности неформального взаимодействия участников образовательного процесса.

Цели разработки сервиса состоят в следующем:

1) расширение функционала неформального взаимодействия участников образовательного процесса с помощью текстовых сообщений, совместных обсуждений, аудио и видео звонков;

2) повышение успеваемости за счёт внедрения новых способов поддержки студентов и оказания помощи в решении различных проблем и вопросов;

3) повышение уровня коммуникации и сотрудничества с помощью удобного обмена сообщениями и ресурсами, обсуждений и функций для совместной работы участников образовательного процесса университета.

Таким образом, основная цель разработки сервиса неформального взаимодействия участников образовательного процесса ВУЗа – это создание уникальной информационной системы, способной предоставить наиболее эффективный и полезный функционал для общения студентов с преподавателями.

1.3.2 Классы и роли пользователей сервиса

Для сервиса выделяются следующие классы пользователей:

- студенты;
- преподаватели;
- модераторы;
- администраторы.

Класс студентов включает следующие роли:

- активные участники;
- пассивные участники.

В качестве активных участников студенты могут управлять своей страницей, писать сообщения, обмениваться информацией и файлами с другими пользователями, создавать обращения на «Хелп-доске» и т. д.

Студенты как пассивные участники используют сервис для получения необходимых материалов, актуальной образовательной информации, сведения о занятиях и широкий спектр новостей.

Класс преподавателей включает следующие роли:

- поставщики информации;
- наставники.

Преподаватели как поставщики информации имеют возможность создавать групповые обсуждения и управлять ими, выкладывать и редактировать собственные методические и образовательные материалы, предоставлять различную информацию об учебных предметах и курсах.

Преподаватели-наставники предоставляют обратную связь студентам, имеют более высокий статус в личном профиле, отвечают на вопросы студентов в чатах или «Хелп-доске».

Модераторы занимаются мониторингом сообщений и контента на сайте, следят за соблюдением правил сервиса, решают спорные ситуации и отвечают за безопасную и комфортную среду на сервисе.

Администраторы отвечают за общее управление сервисом, включая безопасность и конфиденциальность личных данных, размещение, обновление, верификацию и восстановление контента.

1.3.3 Автоматизируемые бизнес-процессы

Бизнес-процесс – это последовательная, целенаправленная совокупность видов деятельности, в которой за счет управляющего воздействия и ресурсов входы процесса преобразуются в выходы, представляющие ценность для потребителей.

В процессе анализа предметной области выделены следующие автоматизируемые бизнес-процессы, которые будут реализованы в процессе разработки сервиса:

- 1) бизнес-процесс «Процесс общения участников образовательного процесса в мессенджере»;
- 2) бизнес-процесс «Оказание помощи по запросу студента»;
- 3) бизнес-процесс «Организация хранения файлов и их обмена между пользователями»;
- 4) бизнес-процесс «Управление личной учетной записью».

Структура бизнес-процессов и взаимодействие между ними удобно представлять в виде диаграммы деятельности UML с разделами (partitions) или плавательными дорожками (swim lanes) [7]. Данный вариант диаграммы деятельности позволяет показать, какие операции выполняет тот или иной класс пользователей сервиса.

На диаграмме деятельности бизнес-процесса «Процесс общения участников образовательного процесса в мессенджере» (рисунок 5) имеется два действующих лица – Студент и Преподаватель. Студент выполняет вход на сайт и открывает вкладку «Сообщения». Далее он может написать сообщение пользователю или ответить на новое. При отправке сообщения студент выбирает другого студента или преподавателя. Все инициативно отправленные сообщения или в форме ответа перенаправляются соответствующему участнику переписки.

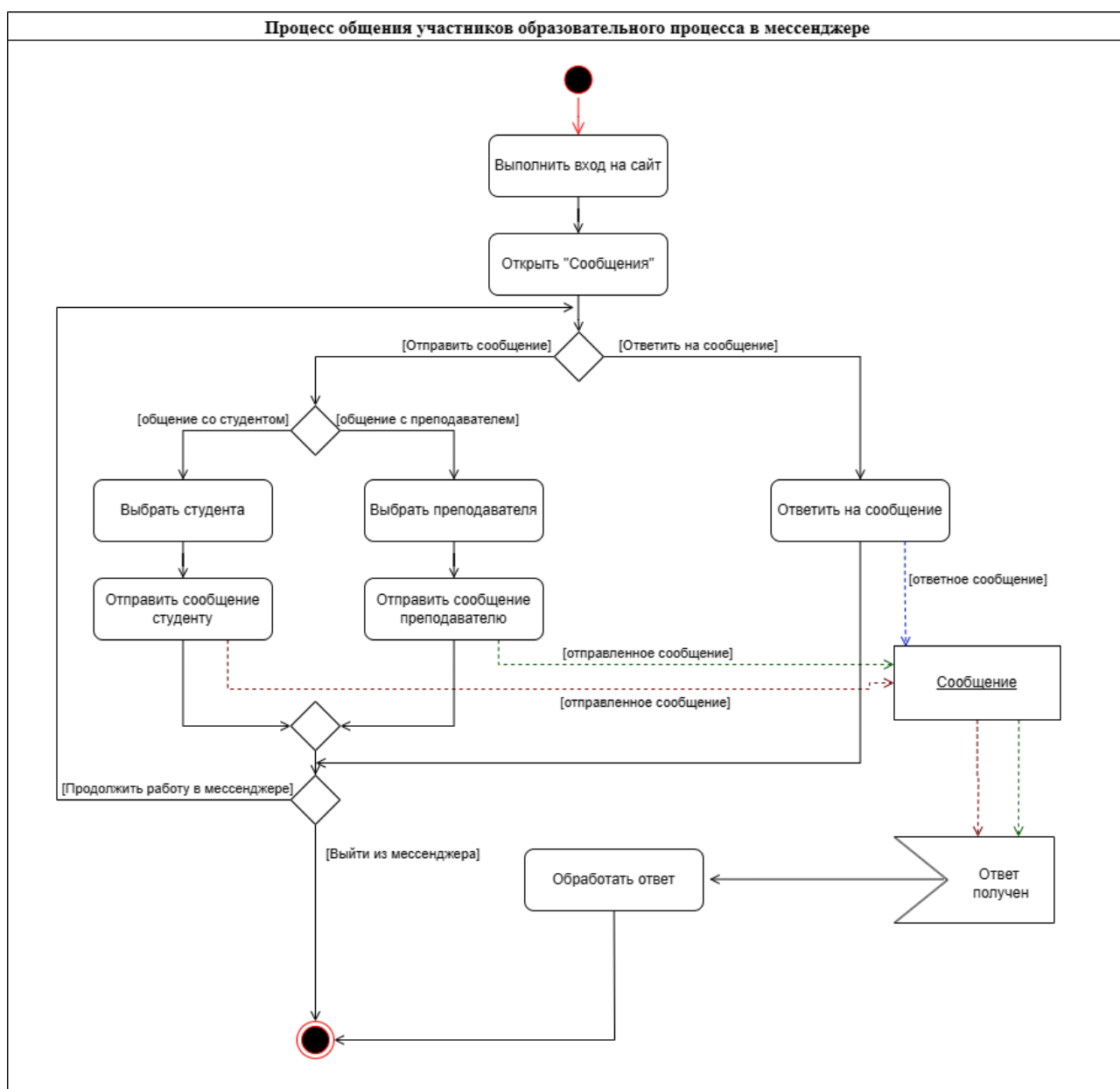


Рисунок 5 – Диаграмма деятельности бизнес-процесса «Общение студентов и преподавателей в личном чате»

Получив ответ от собеседника, студент может либо продолжить, либо прекратить общение с пользователями.

Основные действия преподавателя аналогичны действиям студента, он также может отправлять и получать сообщения от пользователей, продолжить или завершить общение.

На рисунке 6 представлена диаграмма деятельности бизнес-процесса «Оказание помощи по запросу студента».

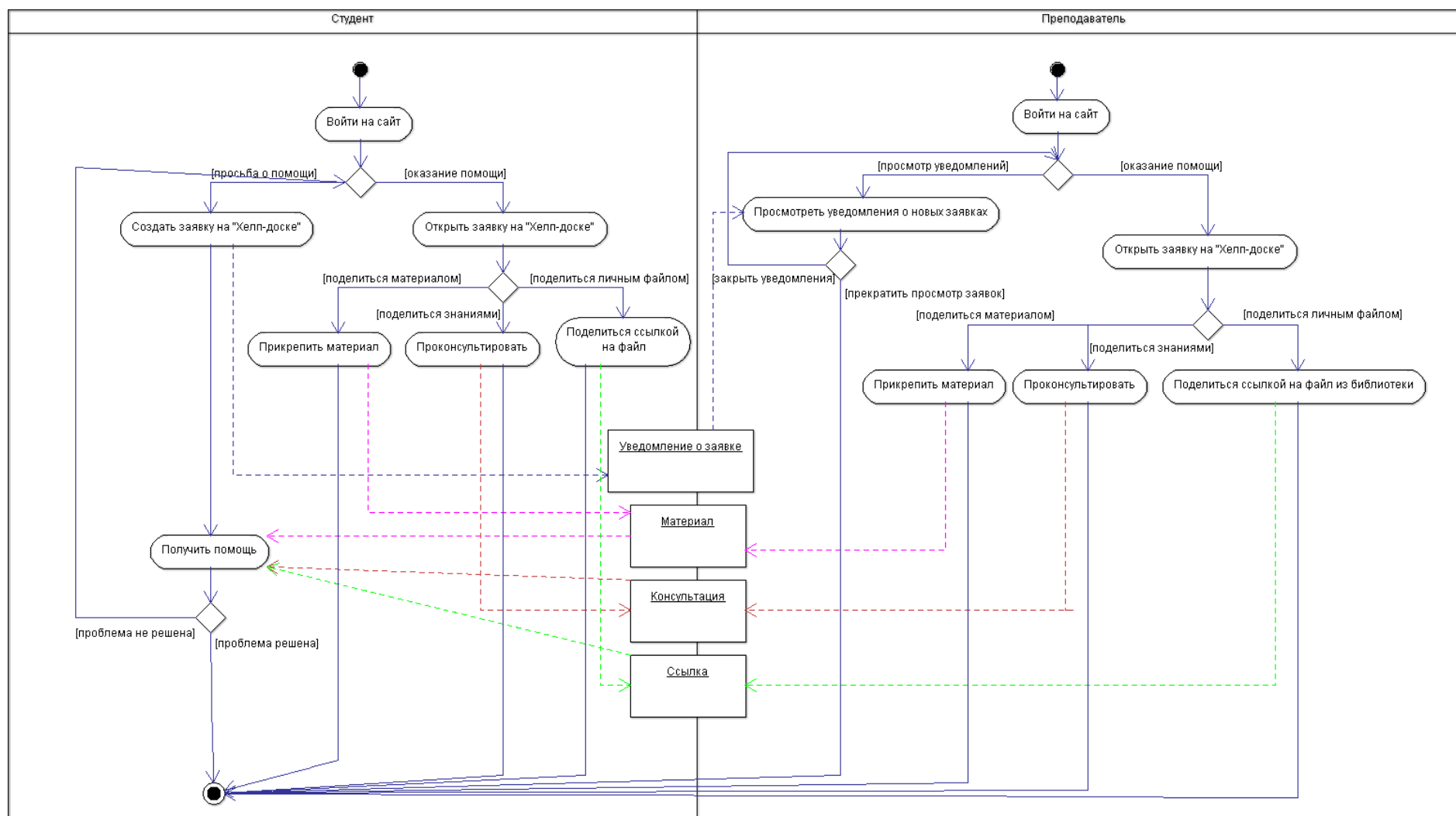


Рисунок 6 – Диаграмма деятельности бизнес-процесса «Оказание помощи по запросу студента»

Студент выполняет авторизацию на сайте и может запросить помощь или её оказать. Для выполнения первой возможной функции он создаёт заявку о проблеме или вопросе на «Хелп-доске». В это время преподавателю приходит уведомление о новой заявке. Он оказывает помощь пользователю в определённом формате. Если предложенное решение помогло, заявка закрывается. Иначе студент в том же самом обсуждении создаёт ещё одну «просьбу», пользователи ему помогают, и данные действия выполняются до тех пор, пока проблема не будет решена.

Преподаватель, выполнив авторизацию на сайте, может либо посмотреть уведомления о заявках, либо сразу начать решать проблему студента. Оказать помощь он может в следующем виде: прикрепить учебный или неучебный материал, проконсультировать или что-то посоветовать студенту, поделиться ссылкой на файл из своей библиотеки. Если проблема не была решена, то преподаватель предлагает дополнительную помощь.

Студент также может оказывать поддержку своим коллегам: для этого у него представлен такой же набор способов оказать помощь, как и у преподавателя.

1.4 Планирование процесса разработки сервиса

Для успешной разработки программного обеспечения требуется его тщательное планирование. Грамотный подход к организации процессов позволяет оптимизировать использование ресурсов, снизить риски, улучшить качество продукта и повысить удовлетворенность конечных пользователей [8]. Именно поэтому планирование процесса разработки сервиса неформального взаимодействия участников образовательного процесса ВУЗа является первостепенной задачей на старте проекта.

В качестве инструмента планирования был выбран Microsoft Project 2013. Данный инструмент помогает правильно распланировать ресурсы и чётко контролировать выполнение задач. Лист ресурсов, необходимых для разработки сервиса, представлен на рисунке 7.

	Название ресурса	Тип	Единицы	Краткое	Группа	Макс.	Стандартная	Ставка	Затраты на	Начисление	Базовый
1	Программист	Трудовой		П		100%	350,00 Р/ч	450,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
2	Аналитик	Трудовой		А		100%	150,00 Р/ч	200,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
3	Тестировщик	Трудовой		Т		100%	200,00 Р/ч	250,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
4	Интернет	Затраты		И						Пропорциональное	
5	Лицензия ПС	Затраты		Л						Пропорциональное	
6	Компьютер	Материальный		К			47 000,00 Р		0,00 Р	Пропорциональное	

Рисунок 7 – Ресурсы проекта

В процессе планирования необходимо выделить основные задачи, подзадачи, длительность их выполнения, а также распределить ресурсы по задачам. В Microsoft Project существует таблица с общим представлением данных (рисунок 8).

	Название задачи	Длительность	Начало	Завершение	При	Названия ресурсов
1	Начало проекта	0 дней	Пн 25.03.24	Пн 25.03.24		
2	Анализ предметной области и постановка задачи на разработку	12 дней	Пн 25.03.24	Вс 07.04.24		
3	Анализ предметной области	3 дней	Пн 25.03.24	Ср 27.03.24	1	Аналитик; Интернет[200,00 Р]; Компьютер[1]; Электричество[120,00 Р]
4	Анализ приложений-аналогов	2 дней	Чт 28.03.24	Пт 29.03.24	3	Аналитик; Интернет[200,00 Р]; Компьютер[1]; Электричество[80,00 Р]
5	Анализ процессов управления метаданными	3 дней	Вс 31.03.24	Вт 02.04.24	4	Аналитик
6	Определение перечня автоматизируемых бизнес-процессов	1 день	Ср 03.04.24	Ср 03.04.24	5	Аналитик; Компьютер[1]
7	Постановка задачи на разработку сервиса	3 дней	Чт 04.04.24	Вс 07.04.24	6	Аналитик; Бумага[1]; Канцелярские принадлежности[1]
8	Анализ предметной области и постановка задачи завершены	0 дней	Вс 07.04.24	Вс 07.04.24	7	
9	Разработка требований к программным средствам сервиса	14 дней	Пн 08.04.24	Вт 23.04.24	2	
10	Моделирование функциональных возможностей сервиса	3 дней	Пн 08.04.24	Ср 10.04.24	8	Программист; Компьютер[1]; Электричество[120,00 Р]; Бумага[1]; Канцеляр
11	Разработка требований к базе данных	2 дней	Чт 11.04.24	Пт 12.04.24	10	Программист; Компьютер[1]; Электричество[80,00 Р]
12	Разработка требований к архитектуре	3 дней	Вс 14.04.24	Вт 16.04.24	11	Программист; Компьютер[1]; Электричество[80,00 Р]
13	Разработка требований к пользовательскому интерфейсу	3 дней	Ср 17.04.24	Пт 19.04.24	12	Программист; Компьютер[1]; Электричество[120,00 Р]
14	Разработка требований к аппаратно-программному окружению	2 дней	Вс 21.04.24	Пн 22.04.24	13	Программист; Компьютер[1]; Электричество[80,00 Р]
15	Обоснование выбора средств разработки	1 день	Вт 23.04.24	Вт 23.04.24	14	Программист; Компьютер[1]; Электричество[40,00 Р]
16	Разработка требований завершена	0 дней	Вт 23.04.24	Вт 23.04.24	15	
17	Разработка серверных программных средств сервиса	31 дней	Ср 24.04.24	Ср 29.05.24	9	
18	Разработка программной архитектуры сервиса	7 дней	Ср 24.04.24	Ср 01.05.24	13	Программист; Компьютер[1]; Электричество[280,00 Р]; Интернет[500,00 Р]
19	Разработка пользовательского интерфейса	4 дней	Чт 02.05.24	Вт 07.05.24	18	Программист; Компьютер[1]; Электричество[200,00 Р]; Интернет[500,00 Р]
20	Разработка базы данных	3 дней	Ср 08.05.24	Пн 13.05.24	19	Программист; Компьютер[1]; Электричество[280,00 Р]; Интернет[500,00 Р]
21	Разработка серверного программного обеспечения	8 дней	Вт 14.05.24	Пт 24.05.24	20	Программист; Компьютер[1]; Электричество[400,00 Р]; Интернет[600,00 Р]
22	Разработка клиентского программного обеспечения	4 дней	Вс 26.05.24	Ср 29.05.24	21	Программист; Компьютер[1]; Электричество[120,00 Р]; Интернет[300,00 Р]
23	Разработка серверных программных средств завершена	0 дней	Ср 29.05.24	Ср 29.05.24	22	
24	Контроль качества программного обеспечения сервиса	8 дней	Чт 30.05.24	Пт 07.06.24	17	
25	Планирование процесса контроля качества	2 дней	Чт 30.05.24	Пт 31.05.24	24	Тестировщик; Компьютер[1]; Электричество[80,00 Р]
26	Тестирование серверных программных средств	4 дней	Вт 04.06.24	Пт 07.06.24	25	Тестировщик; Компьютер[1]; Электричество[160,00 Р]
27	Оценка качества программного кода	2 дней	Вс 02.06.24	Пн 03.06.24	25	Компьютер[1]; Тестировщик; Электричество[80,00 Р]
28	Контроль качества завершён	0 дней	Пт 07.06.24	Пт 07.06.24	26; 27	
29	Завершение работы над проектом	5 дней	Пн 10.06.24	Пт 14.06.24	24	
30	Оформление документации	4 дней	Пн 10.06.24	Чт 13.06.24	28	Интернет[200,00 Р]; Бумага[1]; Канцелярские принадлежности[1]; Аналитик
31	Сдача проекта	1 день	Пт 14.06.24	Пт 14.06.24	30	Программист; Интернет[100,00 Р]; Компьютер[1]; Электричество[40,00 Р]
32	Проект завершен	0 дней	Пт 14.06.24	Пт 14.06.24	31	
33	Конец проекта	0 дней	Пт 14.06.24	Пт 14.06.24	32	

Рисунок 8 – График выполнения задач

Для наглядного представления плана разработки была создана диаграмма Ганта (рисунок 9).

Диаграмма Ганта – это визуальное представление графика работ, построенное согласно плану проекта. На ней отражены задачи и последовательность их выполнения. Диаграмма Ганта состоит из временной

шкалы, расположенной на горизонтальной оси, и списка задач на вертикальной оси [9]. По середине находится срок, за который нужно выполнить определённую задачу.

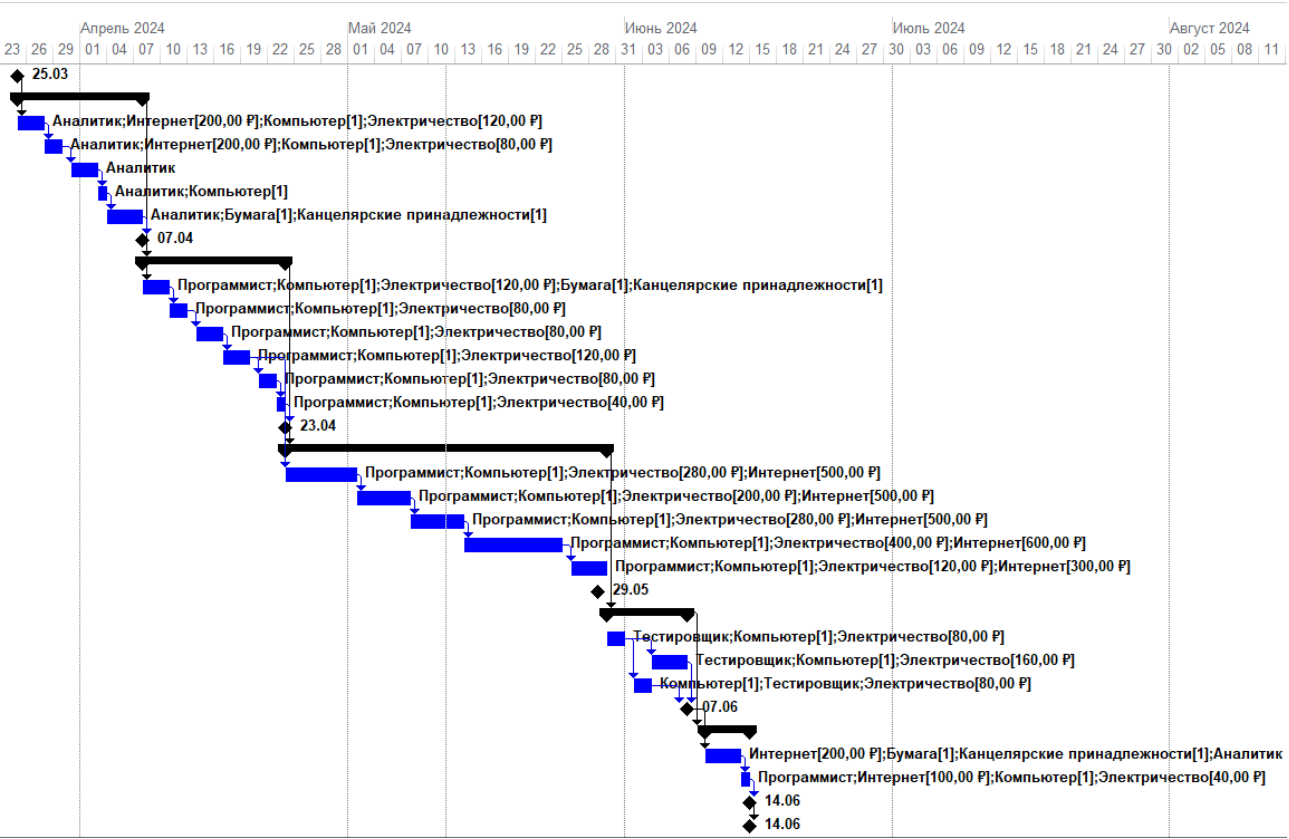


Рисунок 9 – Диаграмма Ганта

Для того чтобы проанализировать и увидеть общую картину результатов планирования разработки сервиса, в Microsoft Project существуют отчёты. В зависимости от того, что необходимо отобразить в отчёте, есть отчёты по ресурсам, затратам, ходу выполнения работы и другие. В данном случае был выбран отчёт панели мониторинга «Обзор затрат» (рисунок 10). Он включает в себя все, оставшиеся затраты, их состояние и ход выполнения задач в сравнении с затратами по времени.

Из отчёта видно, что финансовые затраты на разработку сервиса неформального взаимодействия участников образовательного процесса ВУЗа в совокупности составили 389 360,00 рублей, длительность проекта, с учетом нерабочих дней составила 82 дня.

ОБЗОР ЗАТРАТ

ПН 25.03.24 - ПТ 14.06.24

ЗАТРАТЫ

389 360,00 Р

ОСТАВИШИЕСЯ ЗАТРАТЫ

389 360,00 Р

% ЗАВЕРШЕНИЯ

0%

ХОД ВЫПОЛНЕНИЯ И ЗАТРАТЫ
Ход выполнения в сравнении с затратами во времени. Превышение значения в строке "Совокупные затраты" над значением в строке "Процент завершения" указывает на возможное превышение бюджета.



СОСТОЯНИЕ ЗАТРАТ
Состояние затрат для задач верхнего уровня.

Название	Фактически затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало проекта	0,00 Р	0,00 Р	0,00 Р	0,00 Р	0,00 Р
Анализ предметной области и постановка задачи на разработку	0,00 Р	62 900,00 Р	0,00 Р	62 900,00 Р	62 900,00 Р
Разработка требований к программным средствам сервиса	0,00 Р	87 620,00 Р	0,00 Р	87 620,00 Р	87 620,00 Р
Разработка серверных программных средств сервиса	0,00 Р	122 880,00 Р	0,00 Р	122 880,00 Р	122 880,00 Р
Контроль качества программного обеспечения сервиса	0,00 Р	60 120,00 Р	0,00 Р	60 120,00 Р	60 120,00 Р
Завершение работы над проектом	0,00 Р	55 840,00 Р	0,00 Р	55 840,00 Р	55 840,00 Р
Конец проекта	0,00 Р	0,00 Р	0,00 Р	0,00 Р	0,00 Р

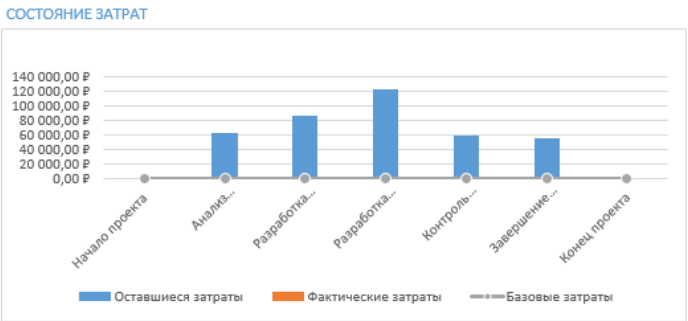


Рисунок 10 – Отчёт «Обзор затрат»

На основании анализа плана реализации и данных обзора затрат, необходимых для выпуска готового проекта, можно сделать вывод, что разработка программного сервиса неформального взаимодействия участников образовательного процесса ВУЗа является экономически выгодным.

2 Проектирование сервиса для неформального взаимодействия участников образовательного процесса ВУЗа

2.1 Моделирование функциональных возможностей сервиса

Одним из этапов проектирования сервиса для неформального взаимодействия участников образовательного процесса ВУЗа является моделирование разрабатываемого функционала. Для отображения всех функциональных возможностей сервиса были построены диаграмма вариантов использования (или диаграмма прецедентов) для актёров «Студент» и «Преподаватель» (рисунок 11) и диаграмма вариантов использования для актёра «Пользователь», отображающая общий функционал Студента и Пользователя (рисунок 12).

Прецеденты – это технология определения функциональных требований к системе. Работа прецедентов заключается в описании типичных взаимодействий между пользователями системы и самой системой и предоставлении описания процесса ее функционирования.

В терминах прецедента пользователи называются актерами. Актер (actor) представляет собой некую роль, которую пользователь играет по отношению к системе. Актёры действуют в рамках прецедентов. Один актер может выполнять несколько прецедентов; и наоборот, в соответствии с одним прецедентом могут действовать несколько актеров [11].

Для наглядности на диаграммах бежевым цветом выделены варианты использования, показывающие операции с данными, розовым – бизнес-операции, белым – бизнес-процессы.

На рисунке 11 показаны индивидуальные функциональные возможности актёров «Студент» и «Преподаватель». Студент может разместить запрос и обработать отклики на него, то есть взаимодействовать с ответами на опубликованную проблему. Преподаватель может помочь с решением проблемы. Для перехода к запросу он может воспользоваться двумя способами: через уведомление и через раздел «Help-доска».

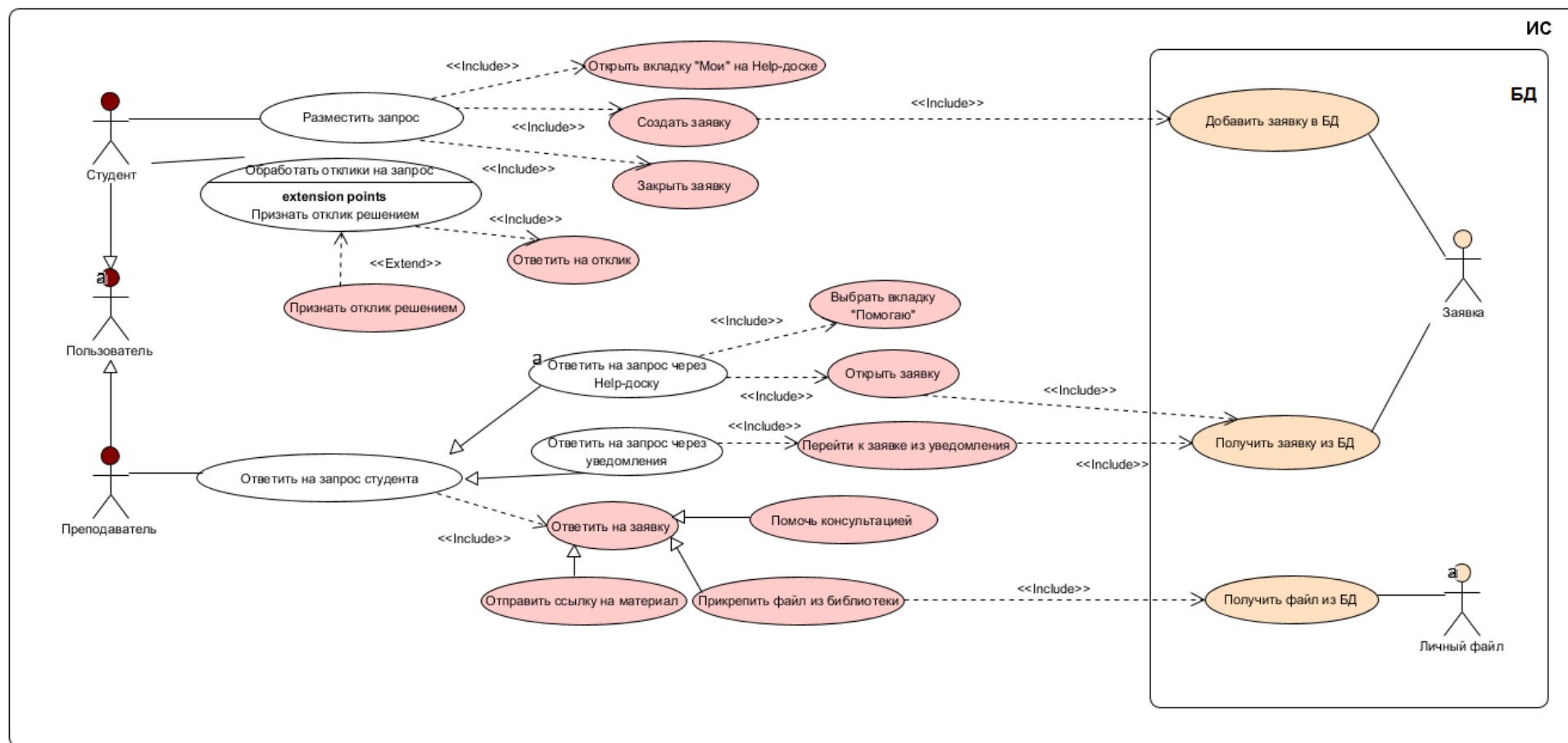


Рисунок 11 – Диаграмма вариантов использования для актёров «Студент» и «Преподаватель»

В таблице 3 представлены спецификации актеров представленной диаграммы.

Таблица 3 – Спецификации актеров-пользователей

№ п/п	Актер	Роль	Краткое описание
1	Студент	Класс пользователя	Учащийся ВУЗа, который активно участвует в образовательном процессе и в ходе обучения задаёт различные учебные вопросы и общается с другими студентами и преподавателями
2	Преподаватель	Класс пользователя	Сотрудник ВУЗа, который активно участвует в образовательном процессе и в ходе преподавания помогает решить различные проблемы студентов
3	Пользователь	Класс пользователя	Лицо, которое пользуется сервисом
4	Заявка	Класс информационных объектов	Сообщение от студента, содержащее в себе намерение получить помощь от других пользователей

Ниже, в таблице 4, представлены краткие спецификации вариантов использования. Спецификация – это описание последовательности действий пользователя в системе в рамках одного варианта использования.

Таблица 4 – Краткие спецификации ВИ актёров «Студент» и «Преподаватель»

Актёр			
№ п/п	Вариант использования	Уровень	Краткое описание
Студент			
1	Разместить запрос	Бизнес-процесс	Студент открывает раздел «Help-доска», создаёт заявку, которая формируется системой.
2	Обработать отклики на запрос	Бизнес-процесс	Студент отвечает на отклик по запросу и при желании признаёт ответ решением.
3	Открыть вкладку «Мои» на Help-доске	Бизнес-операция	Студент открывает вкладку «Мои» на Help-доске.
4	Создать заявку	Бизнес-операция	Студент создаёт заявку на Help-доске.
5	Заккрыть заявку	Бизнес-операция	Студент после создания заявки её закрывает.
6	Ответить на отклик	Бизнес-операция	Студент отвечает на отклик по запросу.

Продолжение таблицы 4

Актёр			
№ п/п	Вариант использования	Уровень	Краткое описание
Студент			
7	Признать отклик решением	Бизнес-операция	Студент признаёт ответ решением, если он таковым является.
8	Добавить заявку в БД	Операция с данными	При добавлении студентом заявки, она автоматически заносится в базу данных.
Преподаватель			
1	Ответить на запрос студента	Бизнес-процесс	Преподаватель открывает заявку определённым способом и отвечает на неё одним из видов помощи.
2	Ответить на запрос через Help-доску	Бизнес-процесс	Преподаватель открывает заявку через вкладку «Помогаю» на Help-доске и отвечает на неё одним из видов помощи.
3	Ответить на запрос через уведомления	Бизнес-процесс	Преподаватель открывает заявку через полученное уведомление о ней и отвечает на не одним из видов помощи.
4	Выбрать вкладку «Помогаю»	Бизнес-операция	Преподаватель при ответе на запрос с Help-доски для открытия заявки выбирает на доске вкладку «Помогаю».
5	Открыть заявку	Бизнес-операция	Преподаватель открывает саму заявку на вкладке «Помогаю».
6	Перейти к заявке из уведомления	Бизнес-операция	Преподаватель при ответе на запрос с Help-доски переходит к заявке из уведомления о ней.
7	Помочь консультацией	Бизнес-операция	Способ ответа преподавателя на заявку в виде консультации.
8	Ответить на заявку	Бизнес-операция	Преподаватель открывает заявку и отвечает на неё определённым способом.
9	Отправить ссылку на материал	Бизнес-операция	Способ ответа преподавателя на заявку в виде ссылки на различный материал.
10	Прикрепить файл из библиотеки	Бизнес-операция	Способ ответа преподавателя на заявку в виде прикрепленного файла из своей библиотеки.
11	Получить заявку из БД	Операция с данными	При переходе к заявке система получает заявку из базы данных.
12	Получить файл из БД	Операция с данными	При прикреплении личного файла система получает из базы данных этот файл.

В таблице 5 приведено полное описание сценария варианта использования «Ответить на запрос студента». Данный сценарий описывает

последовательность возможных действий преподавателя на запрос о помощи студента.

Таблица 5 – Сценарий варианта использования «Ответить на запрос студента»

Прецедент: Ответить на запрос студента
ID: 1
Краткое описание: Преподаватель открывает заявку определённым способом и отвечает на неё одним из видов помощи.
Главные актёры: Преподаватель
Второстепенные актёры: Студент, Заявка, Личный файл
Предусловия: 1. Студент отправляет запрос преподавателю.
Основной поток: 1. Прецедент начинается после того, как студент отправляет запрос, и преподаватель решает на него ответить. 2. ЕСЛИ Преподаватель хочет ответить на запрос через Help-доску 2.1. Преподаватель переходит в раздел «Help-доска». 2.2. Преподаватель выбирает вкладку «Помогаю» на Help-Доске. 2.3. Преподаватель выбирает заявку и открывает её. 3. ЕСЛИ Преподаватель хочет ответить на запрос через уведомления 3.1. Преподаватель открывает уведомления. 3.2. Преподаватель выбирает заявку и переходит к ней из уведомления. 4. Система получает из базы данных заявку и открывает преподавателю. 5. ЕСЛИ Преподаватель хочет помочь консультацией 5.1 Преподаватель в ответе консультирует студента. 6. ЕСЛИ Преподаватель хочет помочь своим материалом 6.1. Система получает из базы данных личные файлы и предоставляет Преподавателю. 6.2. Преподаватель выбирает файл. 6.3. Преподаватель в ответе прикрепляет файл из библиотеки. 7. ЕСЛИ Преподаватель хочет помочь сторонними материалами 7.1. Преподаватель в ответе прикрепляет ссылку на материал. 8. Преподаватель нажимает на кнопку отправления. 9. Система сохраняет ответ Преподавателя.
Постусловия: 1. Ответ Преподавателя на запрос Студента отправлен. 2. Ответ отображается в ветке обсуждения.
Альтернативные потоки: 9. Произошла ошибка отправления сообщения.

В данной диаграмме актёры «Студент» и «Преподаватель» наследуются от актёра «Пользователь». Диаграмма вариантов использования Пользователя представлена на рисунке 12. Она показывает общий функционал всех пользователей сервиса.

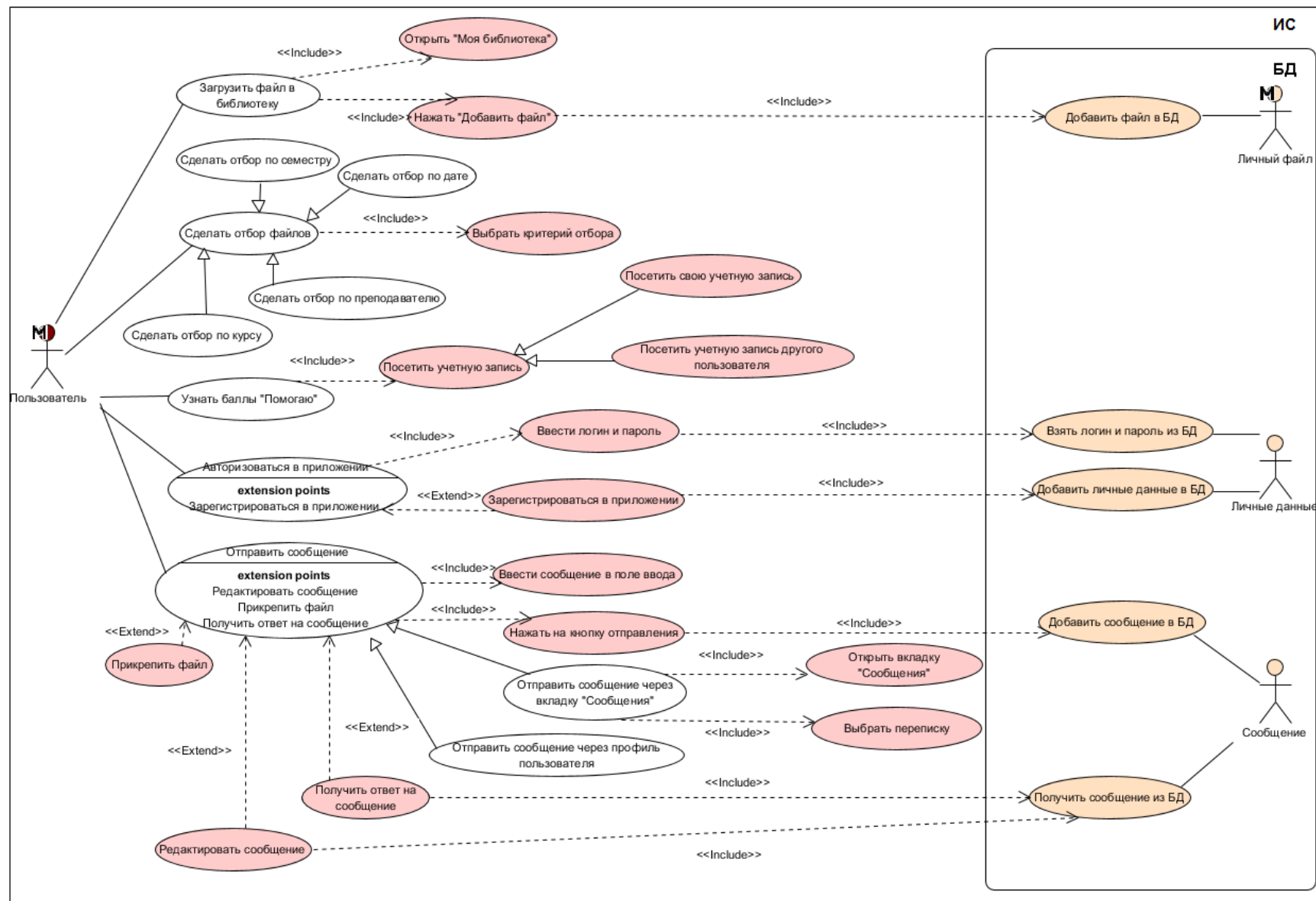


Рисунок 12 – Диаграмма вариантов использования для актёра «Пользователь»

В таблице 6 представлены краткие спецификации вариантов использования.

Таблица 6 – Краткие спецификации ВИ актёра «Пользователь»

Тип вариантов использования			
№ п/п	Вариант использования	Уровень	Краткое описание
Бизнес-процессы			
1	Загрузить файл в библиотеку	Бизнес-процесс	Пользователь открывает личную библиотеку и добавляет файл в коллекцию
2	Сделать отбор файлов	Бизнес-процесс	Пользователь выбирает критерий отбора и фильтрует свои файлы
3	Сделать отбор по семестру	Бизнес-процесс	Отбор личных файлов по критерию «Семестр»
4	Сделать отбор по дате	Бизнес-процесс	Отбор личных файлов по критерию «Дата»
5	Сделать отбор по преподавателю	Бизнес-процесс	Отбор личных файлов по критерию «Преподаватель»
6	Сделать отбор по курсу	Бизнес-процесс	Отбор личных файлов по критерию «Курс»
7	Узнать баллы «Помогаю»	Бизнес-процесс	Пользователь посещает свою или чью-то учетную запись для просмотра баллов помощи на Help-доске
8	Авторизоваться в приложении	Бизнес-процесс	Пользователь вводит логин и пароль и авторизуется в приложении
9	Отправить сообщение	Бизнес-процесс	Пользователь выбирает собеседника, вводит сообщение и отправляет его
10	Отправить сообщение через вкладку «Сообщения»	Бизнес-процесс	Пользователь открывает вкладку «Сообщения», выбирает конкретный чат и отправляет туда сообщение
11	Отправить сообщение через профиль пользователя	Бизнес-процесс	Пользователь выбирает профиль пользователя и отправляет ему сообщение
Бизнес-операции			
1	Открыть «Моя библиотека»	Бизнес-операция	Пользователь открывает раздел «Моя библиотека»
2	Нажать «Добавить файл»	Бизнес-операция	Пользователь выбирает файл и нажимает на кнопку «Добавить файл» для добавления его в библиотеку
3	Выбрать критерий отбора	Бизнес-операция	Пользователь для отбора своих файлов выбирает критерий, по которому будет происходить фильтрация
4	Посетить учетную запись	Бизнес-операция	Пользователь нажимает на пользователя или личный кабинет и переходит в учетную запись
5	Посетить свою учетную запись	Бизнес-операция	Пользователь нажимает вкладку «Личный кабинет» и переходит в личный профиль

Продолжение таблицы 6

Тип вариантов использования			
№ п/п	Вариант использования	Уровень	Краткое описание
6	Посетить учетную запись другого пользователя	Бизнес-операция	Пользователь нажимает на пользователя и переходит в его профиль
7	Ввести логин и пароль	Бизнес-операция	Пользователь для авторизации вводит личные логин и пароль
8	Зарегистрироваться в приложении	Бизнес-операция	Пользователь при отсутствии авторизации регистрируется на сайте
9	Ввести сообщение в поле ввода	Бизнес-операция	Пользователь при отправлении сообщения вводит его в поле ввода
10	Нажать на кнопку отправления	Бизнес-операция	Пользователь нажимает на кнопку отправления после ввода сообщения
11	Открыть вкладку «Сообщения»	Бизнес-операция	Пользователь открывает вкладку «Сообщения»
12	Выбрать переписку	Бизнес-операция	Пользователь выбирает конкретный чат для написания сообщения
13	Прикрепить файл	Бизнес-операция	Пользователь при отправлении сообщения может прикрепить файл к нему
14	Редактировать сообщение	Бизнес-операция	Пользователь может отредактировать сообщение
Операции с данными			
1	Добавить файл в БД	Операция с данными	При загрузке файла система добавляет выбранный файл в базу данных
2	Взять логин и пароль из БД	Операция с данными	При вводе логина и пароля данные берутся автоматически из базы данных
3	Добавить личные данные в БД	Операция с данными	При регистрации в приложении личные данные заносятся в базу данных
4	Добавить сообщение в БД	Операция с данными	При отправлении сообщения оно добавляется в базу данных
5	Получить сообщение из БД	Операция с данными	При взаимодействии с отправленным сообщением оно возвращается из базы данных

В таблице 7 приведено полное описание сценария варианта использования «Отправить сообщение».

Таблица 7 – Сценарий варианта использования «Отправить сообщение»

Прецедент: Отправить сообщение
ID: 1
Краткое описание: Пользователь выбирает собеседника, вводит сообщение и отправляет его.
Главные актёры: Пользователь

Продолжение таблицы 7

Прецедент: Отправить сообщение
Второстепенные актёры: Сообщение
Предусловия: 1. Пользователь авторизуется на сайте
Основной поток: 1. Прецедент начинается после авторизации на сайте. 2. ЕСЛИ Пользователь выбирает написать сообщение через вкладку «Сообщения». 2.1. Пользователь открывает вкладку. 2.2. Выбирает нужный чат. 3. ЕСЛИ Пользователь выбирает написать сообщение через профиль пользователя. 3.1. Пользователь заходит в профиль пользователя. 3.2. Пользователь выбирает «Написать сообщение». 4. Пользователь вводит сообщение в поле ввода. 5. Пользователь нажимает на кнопку отправления сообщения. 6. Система сохраняет сообщение в базе данных.
Расширения: 4а. Пользователь хочет прикрепить файл. .1: Пользователь нажимает на кнопку прикрепления файлов. .2: Пользователь выбирает файл. 6а. Пользователь хочет отредактировать сообщение. .1: Система предоставляет пользователю сообщение. .2: Пользователь редактирует сообщение. 5а. Пользователю приходит ответ на сообщение. .1: Система получает новое сообщение из БД. .2: Пользователь получает ответ на сообщение.
Постусловия: Пользователь видит отправленное сообщение в чате.
Альтернативные потоки: 6. Произошла ошибка отправления сообщения.

Пользователь при необходимости может повторить действия и отправить ещё раз сообщение или прикрепить несколько файлов. При происхождении ошибки пользователь может попробовать заново отправить сообщение или перезагрузить сайт.

2.2 Разработка концептуальной модели сервиса

На основе анализа предметной области с учетом информационных задач и бизнес-процессов сервиса для неформального взаимодействия участников образовательного процесса ВУЗа была разработана концептуальная модель базы

данных представленная в виде диаграммы классов Object-Relational Mapping (ORM) (рисунок 13).

Диаграмма классов – это диаграмма UML, отображающая структуру иерархии классов системы, атрибутов, методов и связей между ними.

ORM – это технология программирования, которая позволяет работать с базами данных как с классами и связывает их с концепциями объектно-ориентированных языков программирования.

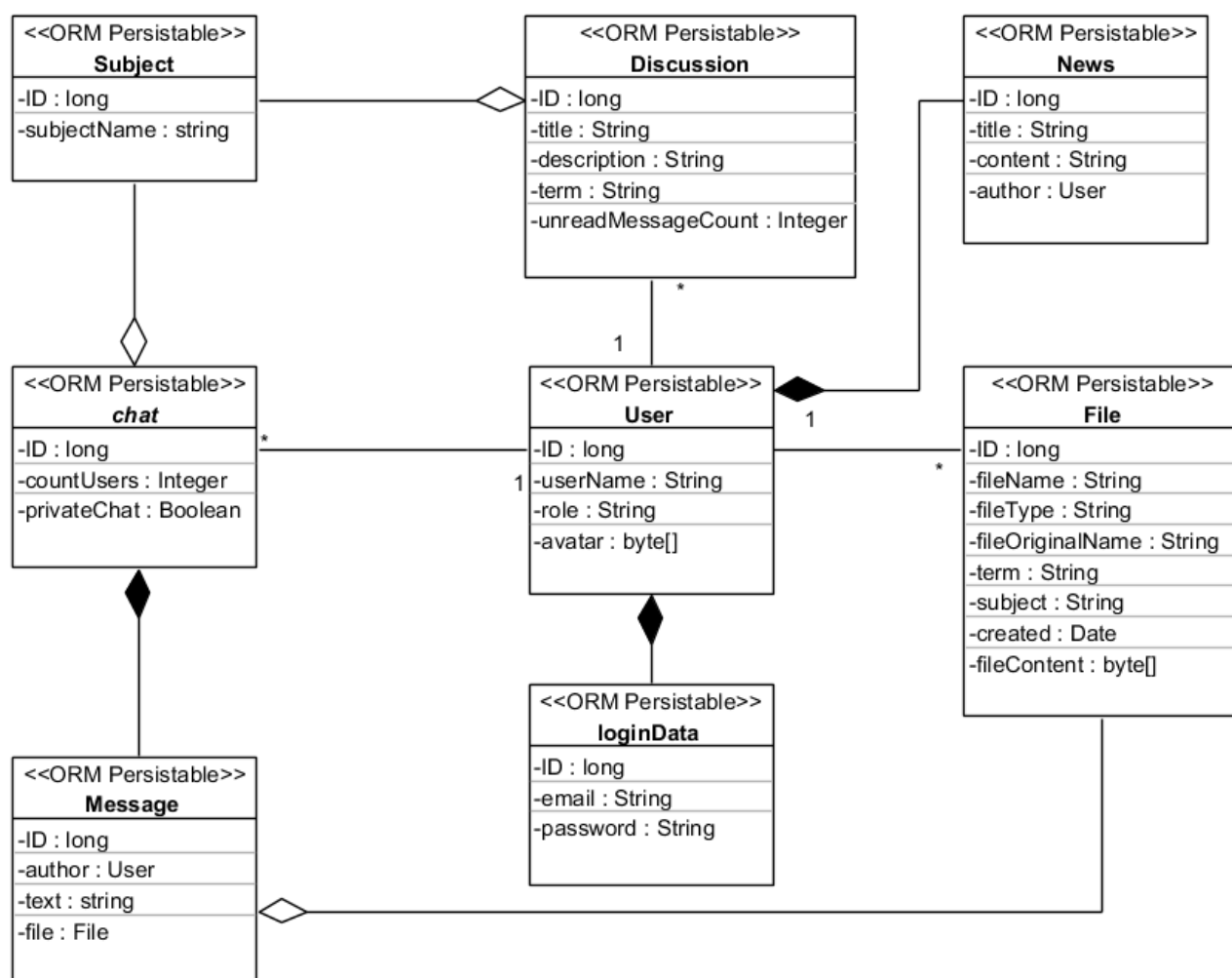


Рисунок 13 – Диаграмма классов сервиса

Классы данной ORM диаграммы являются моделями, на основе которых автоматически будет строится база данных программного сервиса. Всего в ходе моделирования получилось 8 классов.

Описание класов диаграммы классов представлено в таблице 8.

Таблица 8 – Описание классов предметной области

№ п/п	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Авторизация	Данные авторизации пользователя	Стержневой	100000	90% в месяц
2	Пост	Данные поста пользователя	Стержневой	500000	90% в месяц
3	Пользователь	Информация о пользователе	Стержневой	100000	90% в месяц
4	Обсуждение	Данные о запросе пользователя	Стержневой	1000000	60% в месяц
5	Чат	Данные о чатах пользователя	Стержневой	100000	50% в месяц
6	Сообщение	Данные о сообщениях чата	Стержневой	5000000	100% в месяц
7	Файл	Списочный состав личных файлов пользователя	Стержневой	300000	10% в месяц
8	Предмет обсуждения	Списочный состав всех тем для обсуждения	Характеристический	300000	10% в месяц

Полученные результаты являются исходными данными для моделирования базы данных и дальнейшей разработки сервиса. Модель разработана в среде Visual Paradigm 15.2 Enterprise.

2.3 Разработка программной архитектуры сервиса

Для разработки программной архитектуры сервиса была выбрана клиент-серверная архитектура.

Архитектура клиент-сервер – это схема организации вычислительных систем, в которой выполнение задач распределено между клиентом и сервером с помощью сетевого взаимодействия. В этой архитектуре клиент, обычно являющийся пользователем или программой, запрашивает услуги или ресурсы у сервера. Сервер, в свою очередь, отвечает на запросы, предоставляя необходимые данные или функциональность.

Клиент-серверная архитектура обладает следующими преимуществами:

- масштабируемость позволяет равномерно распределять нагрузку между серверами, количество которых можно увеличивать;
- сервер выступает в роли центрального узла, осуществляющего контроль за всей системой, что упрощает её обновление и модификацию;
- повышенная отказоустойчивость системы из-за того, что выход из строя одного компонента не мешает работе других;
- высокая производительность сервера обеспечивает обработку огромного количества запросов от множества пользователей.

В качестве клиента выступает веб-приложение для неформального взаимодействия, которое предоставляет интерфейс взаимодействия сервиса с пользователем. Через него юзер может отправлять запросы по отправлению сообщения, создание заявки о помощи, просмотр личного профиля и так далее.

Сервером является компьютер, который будет предоставлять информацию пользователям и выполнять запросы для пользования сервисом.

На уровне базы данных выступает СУБД, которая выполняет различные операции по работе с данными, хранит всю информацию о пользователе, его файлах, сообщениях, обсуждениях. А также обеспечивает ее целостность и безопасность.

Детально архитектура разрабатываемого сервиса неформального взаимодействия участников ВУЗа показана в виде диаграммы развертывания на рисунке 14.

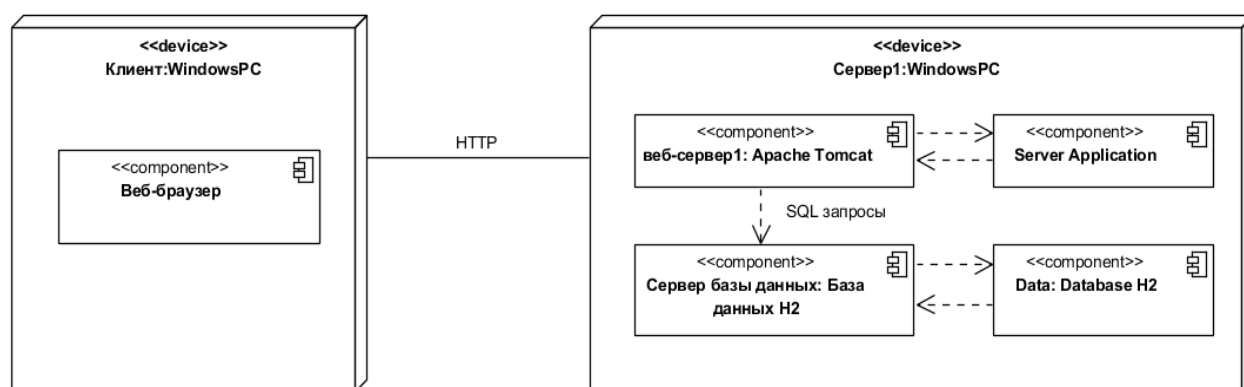


Рисунок 14 – Диаграмма развертывания сервиса

Для представления того, как взаимодействуют между собой архитектурные компоненты, была построена диаграмма последовательности, реализованную для сценария «Разместить заявку на Help-доске» (рисунок 15).

Диаграмма последовательности – это наглядное представление совокупности разных элементов модели системы, изображение того, как и в каком порядке они взаимодействуют [10].

На диаграмме показаны следующие объекты: пользователь, сайт, сервер, база данных. Пользователь нажимает на вкладку «Help-доска». В это время сайт или клиент посылает http-запрос серверу на получение формы Help-доски. Он возвращает данные, а клиент показывает пользователю страницу. Затем пользователь нажимает кнопку обсуждения и получает форму создания запроса таким же образом, как и раздел с доской.

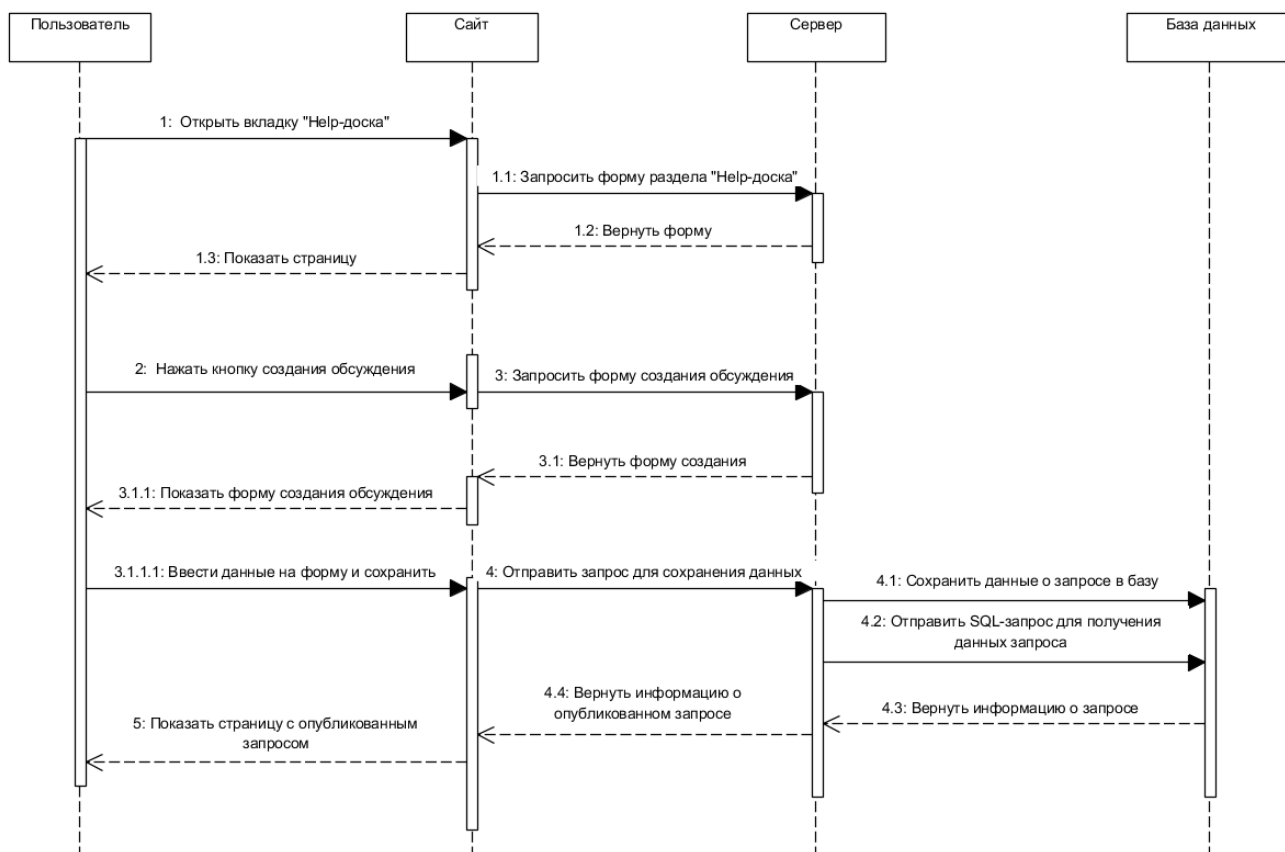


Рисунок 15 – Диаграмма последовательности для взаимодействия компонентов выполнения процесса «Разместить заявку на Help-доске»

После ввода данных и нажатия на кнопку сохранения, клиент отправляет post-запрос серверу для создания нового запроса. Тот, в свою очередь, создаёт запрос SQL для сохранения данных об опубликованной заявке и затем возвращает их для отображения нового обсуждения у пользователя на странице.

2.4 Разработка прототипа пользовательского интерфейса

Важной частью моделирования сервиса для неформального взаимодействия участников ВУЗа является разработка прототипа пользовательского интерфейса.

Пользовательский интерфейс UI (User Interface) – это набор правил и материальные средства, которые обеспечивают взаимодействие системы и пользователя по этим правилам [20].

Существуют следующие виды пользовательского интерфейса:

- командная строка;
- графический интерфейс (WIMP);
- речевой интерфейс (Silk).

Командная строка подразумевает под собой интерфейс, в котором пользователь с помощью клавиатуры вводит разные команды и параметры. Обычно их текст отображается белым цветом на черном окне терминала. После ввода пользователь нажимает на клавиатуре кнопку Enter, и команда выполняется.

На смену командной строке достаточно быстро пришел графический интерфейс (WIMP). Он обладает большим количеством преимуществ, которые связаны с возможностью использования мыши для управления средой и более понятной средой взаимодействия с пользователем.

Аббревиатура WIMP имеет следующую расшифровку: W – Window (окно), I – Image (картинка), M – Menu (меню), P – Pointer (указатель). Это означает, что данный вид интерфейса содержит в себе все графические компоненты для более наглядного и понятного отображения. Он позволяет взаимодействовать с объектами системы при помощи контроллеров (например, мышь или джойстик).

Графический интерфейс используется большинством сайтов и на различных устройствах: компьютерах, смартфонах и планшетах. Именно этот вид интерфейса будет применяться при разработке сервиса.

После того как в различные устройства были внедрены нейронные системы и повысились их вычислительные мощности, была разработана речевая технология пользовательского интерфейса [21]. Пользователь обращается к устройству голосом, тот его принимает, обрабатывает и выдаёт соответствующий результат. К данному виду относятся все виды голосовых помощников, умные колонки, голосовые системы по управлению автомобилем; голосовые элементы, входящие в систему «Умного дома».

В ходе разработки графического интерфейса были созданы шаблоны окон программного сервиса, представленные в приложении Б на рисунках Б.1–Б.7.

Для описания переходов между окнами была построена проводная модель пользовательского интерфейса (рисунок 16).

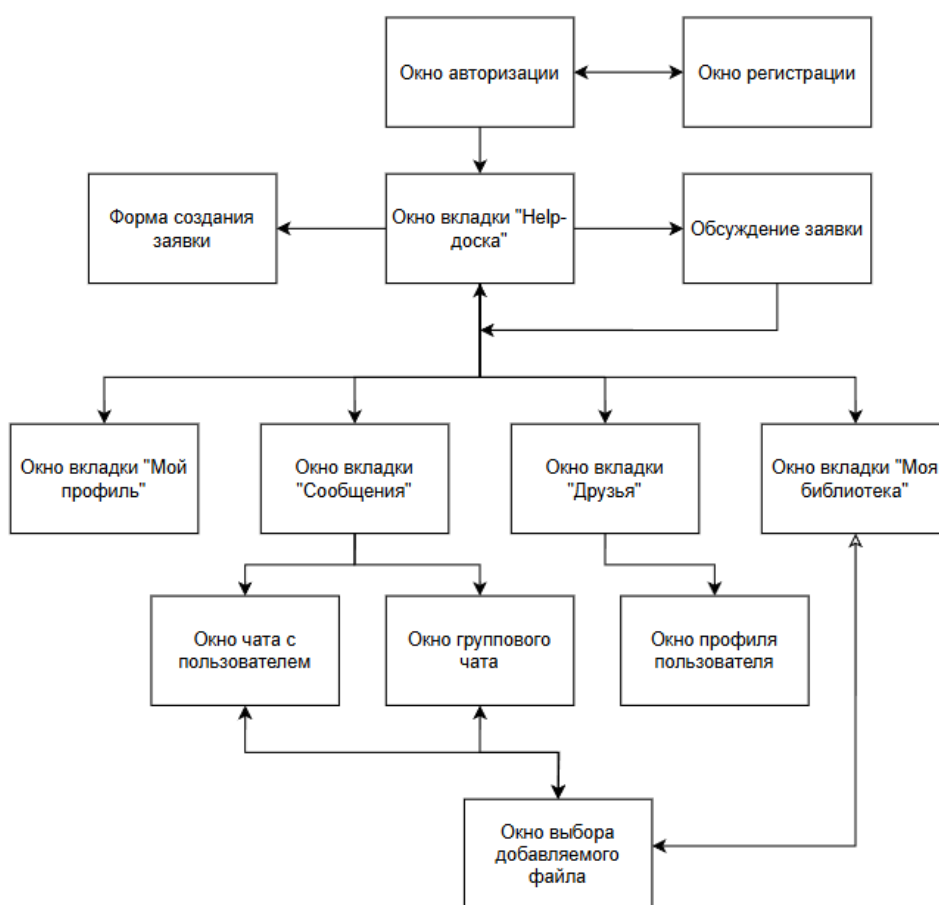


Рисунок 16 – Схема переходов между окнами сервиса

На схеме показано, что работа с пользовательским интерфейсом начинается с появления окна авторизации пользователя. Если он не зарегистрирован и хочет это сделать, то переходит в окно регистрации.

После авторизации перед пользователем открывается окно вкладки «Help-доска», с которой он может перейти во вкладки «Мой профиль», «Сообщения», «Друзья» и «Моя библиотека». Если пользователь хочет создать заявку, ему открывается форма её создания, если хочет открыть уже созданную заявку – форма её обсуждения.

При выборе группового или личного чата на вкладке «Сообщения», у него появляется соответствующее окно. Если он хочет прикрепить файл к сообщению или же добавить файл в «Мою библиотеку», перед ним всплывает окно выбора добавляемого файла.

При взаимодействии с друзьями на вкладке «Друзья», пользователь может перейти к конкретному другу, нажав на него. И тогда перед ним откроется личный профиль данного пользователя.

2.5 Разработка требований к аппаратно-программному окружению

Для разработки сервиса для неформального взаимодействия участников образовательного процесса ВУЗа необходимы следующие минимальные требования к программно-аппаратным средствам:

- 1) JetBrains IntelliJ IDEA версии Community Edition или выше;
- 2) Visual Paradigm Community Edition или Enterprise 15 или выше;
- 3) Apache Tomcat версии 9.0.36 или выше;
- 4) определённые параметры компьютера.

Параметры компьютера состоят из следующих характеристик:

- 1) Microsoft Windows 10, 11;
- 2) 64-разрядная версия Microsoft Windows;
- 3) двухъядерный процессор с тактовой частотой 2 ГГц или лучше;
- 4) 2 ГБ свободной оперативной памяти или 8 ГБ общей оперативной памяти;

- 5) не менее 25 ГБ свободного места на жестком диске;
- 6) разрешение монитора не менее, чем 1024×768.

Важным фактором для разработки сервиса неформального взаимодействия участников образовательного процесса ВУЗа является подключение к глобальной сети Internet.

2.6 Обоснование выбора средств разработки

2.6.1 Обоснование выбора средств автоматизации

В последнее десятилетие всё больше популярности стало набирать новое направление в проектировании информационных систем – автоматизированное проектирование с помощью CASE-средств.

CASE-средства (Computer – Aided Software Engineering) – это методы и технологии, которые позволяют проектировать различные информационные системы (в частности, базы данных) и автоматизировать их создание [13]. CASE-средства позволяют обеспечить удобную и быструю разработку, повысить качество создаваемых БД.

Для обзора средств автоматизации были выбраны следующие CASE-средства:

- Visual Paradigm;
- CA Erwin Data Modeler;
- Rational Rose;
- Case.Аналитик.

CA Erwin Data Modeler – продукт, который является одним из лидеров в области моделирования данных. Он предлагает интуитивно понятный интерфейс для управления комплексными данными. Семейство продуктов CA Erwin Data Modeler включает в себя различные версии, разработанных для удовлетворения всех потребностей пользователей.

Данное CASE-средство поддерживает сразу три нотации: IDEF0, DFD, IDEF3. Огромным преимуществом ERwin DM является графический пользовательский интерфейс, базируемый на операционной системе Windows со

встроенным мощным инструментом для работы с ER-диаграммами (сущность-связь) [14].

К недостаткам продукта можно отнести ограниченные возможности для проведения стоимостного анализа и отсутствие стандартных объектов для описания бизнес-процессов.

Rational Rose представляет собой инструмент автоматизированного проектирования программных средств. Основой продукта является CASE-технология, комплексный подход и использование единой унифицированной нотации на всех этапах жизненного цикла создания программного обеспечения.

Преимущества и недостатки у данного продукта схожи с характеристиками CA Erwin Data Modeler. Также к недостаткам можно отнести высокую цену и сложность освоения продукта. Особенностью Rational Rose является объектно-ориентированный подход. Графические модели, создаваемые с помощью этого средства, основаны на объектно-ориентированных принципах и языке UML.

Case.Аналитик – единственная разработка, направленная именно на отечественный рынок. В основе пакета лежит методология структурного системного анализа Гейне-Сарсоне, обеспечивающая принудительно хороший стиль, предоставление доступа к любой части проекта и другие характеристики [15].

К преимуществам можно отнести минимальные требования к аппаратно-программному обеспечению, возможность экспорта и импорта фрагментов проекта для автономной работы нескольких аналитиков. Из недостатков сложность описания из-за большого количества функций, элементов, сложных взаимосвязей между ними; малое количество совместимых с данным продуктом средств для разработок.

Visual Paradigm – это инструмент, используемый для создания и управления версиями UML или унифицированного языка моделирования, который также поддерживает группу управления объектами, включая нотацию для моделирования бизнес-процессов. Данный продукт способен помочь

пользователю на различных этапах жизненного цикла разработки ПО. В Visual Paradigm есть возможность создания большого количества диаграмм, моделей. Есть функция формирования отчёта на основе данных проекта или отдельной диаграммы. Также к преимуществам можно отнести следующее:

- удобный и понятный интерфейс;
- есть бесплатная 30-дневная версия;
- возможность составления документации у диаграмм;
- создание диаграмм по стандартам UML.

Проанализировав представленные выше CASE-средства, для проектирования сервиса неформального взаимодействия участников образовательного процесса ВУЗа был выбран Visual Paradigm.

2.6.2 Обоснование выбора инструментальных средств разработки программного обеспечения

IDE (Integrated Development Environment) – это интегрированная среда разработки, используемая разработчиками для создания программного обеспечения. IDE предназначены для упрощения разработки программ. Они имеют такие функции, как написание и редактирование кода в одном окне, установка компонентов, тестирование наработок и ещё много другого функционала.

Существуют следующие наиболее популярные IDE:

- IntelliJ IDEA;
- Eclipse;
- NetBeans;
- Microsoft Visual Studio;
- XCode.

Для разработки программного обеспечения была выбрана интегрированная среда IntelliJ IDEA.

IntelliJ IDEA является ведущей IDE для разработки на Java и Kotlin. У неё есть три вида лицензии:

- IntelliJ IDEA Ultimate (платная профессиональная версия);
- IntelliJ IDEA Community Edition (бесплатный вариант для личного и коммерческого использования);
- IntelliJ IDEA EDU (бесплатная версия, созданная специально для студентов и преподавателей).

В данной интегрированной среде возможна разработка не только на языке Java, но и на других языках, работающих на виртуальной машине Java. К ним относятся Kotlin, Scala и Groovy. Также имеется поддержка фреймворков, используемых в веб-разработке.

IntelliJ IDEA обладает широкими возможностями проверки качества и валидности кода с помощью инспекций, которые выполняются «на лету». Инспекции помогают быстрее писать код в соответствии с самыми строгими стандартами качества. Если она обнаруживает ошибки, то сразу на этапе написания кода предлагает их быстрое исправление.

В IDE сильно развита поддержка пользователя. На официальном сайте JetBrains есть подробное с фотографиями описание различных по сложности кейсов. Есть пошаговые инструкции по установке программ, дополнительных компонентов, расширений, а также отладке и тестированию кода.

В целом, интегрированная среда IntelliJ IDEA является мощным инструментом для разработки сервиса с развитой архитектурой, богатым функционалом и понятным интерфейсом.

2.6.3 Обоснование выбора языковых средств разработки

В феврале 2024 года редакция Skillbox.by составила топ языков программирования, основываясь на их опросе среди специалистов, популярности и других критериях [16]. Рейтинг был сформирован исходя из данных, предоставленных следующими редакциями: RedMonk, Stack Overflow, PYPL, TIOBE.

RedMonk – платформа, которая составляет рейтинг языковых средств на основе популярности упоминаний языков на GitHub и Stack Overflow.

Stack Overflow – сервис, который работает в формате «вопрос-ответ». Предназначен для IT-специалистов.

PYPL – рейтинг популярности по версии Google. Из этого источника данные были взяты за февраль 2024 года.

TIOBE – индекс, который является показателем популярности языков программирования. Индекс основан на отзывах большого количества квалифицированных разработчиков и инженеров. Обновляется раз в месяц.

В топ-5 данного рейтинга входят следующие языки программирования:

- 1) Python;
- 2) Языки программирования C и C++;
- 3) Java;
- 4) C#;
- 5) JavaScript.

Python – высокоуровневый язык программирования, разработанный в 1989 году Гуидо ван Россумом. Отличается простотой синтаксиса и лаконичностью кода. Python используют в научных исследованиях, анализе данных, разработке веб-приложений, искусственном интеллекте, машинном обучении и средствах автоматизации. У языка богатая библиотека базовых функций, процедур и обширный перечень пакетов. Но некоторые задачи Python может выполнять более медленно чем компилируемые языковые среды.

Язык программирования C – это процедурный язык программирования, который был разработан в 1969 – 1973 годах сотрудниками Bell Labs Кеном Томпсоном и Деннисом Макалистэйр Ритчи. Изначально был создан для использования в операционной системе Unix, но позже его перенесли на множество других платформ.

В начале 1980-х годов Бьерн Страуструп, также являющийся сотрудником Bell Labs, разработал C++ как объектно-ориентированное расширение языка C. Изначально он добавил к языку C возможность работы с классами и объектами, а в 1983 году после добавления большого количества синтаксических конструкций язык был переименован из C с классами в «C++».

Языки программирования С и С++ предназначены для написания системного ПО и решений, требующих для работы высокой производительности. Инструменты обеспечивают прямой доступ к системным ресурсам, высокую производительность и возможность оптимизации кода. Однако они более сложны в изучении, и программистам начального уровня писать на них несколько сложнее.

В качестве языка программирования, предназначенного для разработки сервиса неформального взаимодействия участников образовательного процесса ВУЗа, был выбран Java.

Java – это объектно-ориентированный язык программирования, который появился в мае 1995 года благодаря компании Sun Microsystems. Изначально был предназначен для написания программ на встроенных системах. Java имеет строгую типизацию, которая не позволяет совмещать в выражениях различные типы и не выполняет автоматически неявные преобразования [17].

Java является кроссплатформенным языком, что позволяет запускать один и тот же код на разных платформах. Такое происходит из-за JVM (Java Virtual Machine), которая является посредником между кодом и железом. Компилятор преобразовывает полученный код в набор байтов, а виртуальная машина, учитывая платформу, на которой запускается программа, загружает байт-код и выполняет его.

Также к преимуществам данного языка относятся надежность, благодаря статической типизации, поддержка множества фреймворков, возможность параллельной разработки и хорошую организацию кода.

Для обращения к данным, хранящимся в базе данных, был использован язык запросов SQL, разработанный в 1970-х годах компанией IBM. Это декларативный язык программирования, предназначенный для манипулирования данными в реляционных базах данных [18]. Данный язык обладает четкой и понятной структурой, что делает его относительно простым для работы с данными.

К достоинствам языка SQL относятся:

- 1) стандартизированность;
- 2) возможность создания интерактивных запросов;
- 3) поддержка клиент-серверной архитектуры;
- 4) независимость от конкретной СУБД;
- 5) реляционная основа языка.

Также с помощью SQL можно управлять доступом к данным, а именно: предоставлять, закрывать или ограничивать в правах пользователей. В дополнение к этому можно выдать им определённый набор функций: чтение, изменение, создание, удаление, копирование. Это позволяет защищать базы данных от взлома или несогласованных изменений.

2.6.4 Обоснование выбора средств управления данными

В ходе выбора наиболее подходящей системы управления базами данных (СУБД) для сервиса неформального взаимодействия участников образовательного процесса ВУЗа были рассмотрены следующие СУБД: MySQL, PostgreSQL, MS SQL Server и H2.

Результаты анализа показали, что для сервиса, разрабатываемого на языке программирования Java, более удобным и эффективным является СУБД H2.

H2 – открытая кроссплатформенная система управления базами данных, полностью написанная на языке Java. Данная СУБД была разработана в декабре 2005 года разработчиком Hupersonic SQL Томасом Мюллером. Она обладает небольшим размером (около 2,5 МБ jar-файла), что делает ее идеальной для встроенных приложений и сред разработки.

H2 – это клиент-серверное приложение, которое может работать полностью в памяти, сохраняя данные на диске в приложении (встроенный режим) или работать как автономный сервер базы данных.

По сравнению с другими СУБД H2 имеет достаточно высокую производительностью [19]. На графике, сгенерированном непосредственно из данных Бенчмарка с официального сайта H2, можно увидеть результаты

сравнения СУБД H2, HSQLDB, Derby, PostgreSQL и MySQL по критерию производительности (рисунок 17).

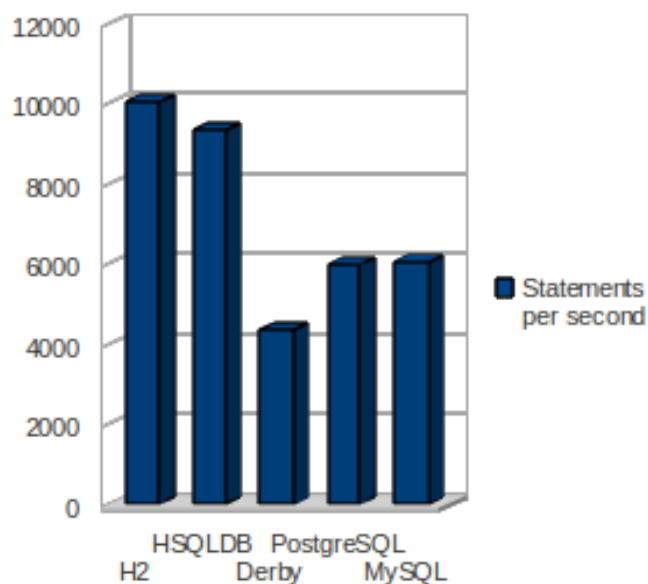


Рисунок 17 – График сравнения производительности СУБД

Таким образом, на выбор СУБД H2 повлияли следующие его характеристики:

- легкая интеграция с Java-приложениями;
- кроссплатформенность;
- высокая производительность;
- очень быстрый JDBC API с открытым исходным кодом;
- компактный размер jar-файла около 2,5 МБ.

СУБД продолжает развиваться и периодически выпускает новые улучшенные версии.

3 Разработка программного обеспечения сервиса для неформального взаимодействия участников образовательного процесса ВУЗа

3.1 Разработка базы данных

Реализация сервиса начинается с разработки базы данных, в которой будет храниться вся информация о приложении.

База данных – это упорядоченный определённым образом набор структурированных данных, которые чаще всего сохраняются в электронном виде в компьютерной системе. Базы данных управляются системой управления базами данных (СУБД). К базе можно обращаться с целью редактирования, удаления или добавления данных. Для этого в большинстве баз используется язык запросов SQL. Хранение и обработка данных сервиса будет происходить в СУБД H2.

Список таблиц базы данных представлен на рисунке 18.

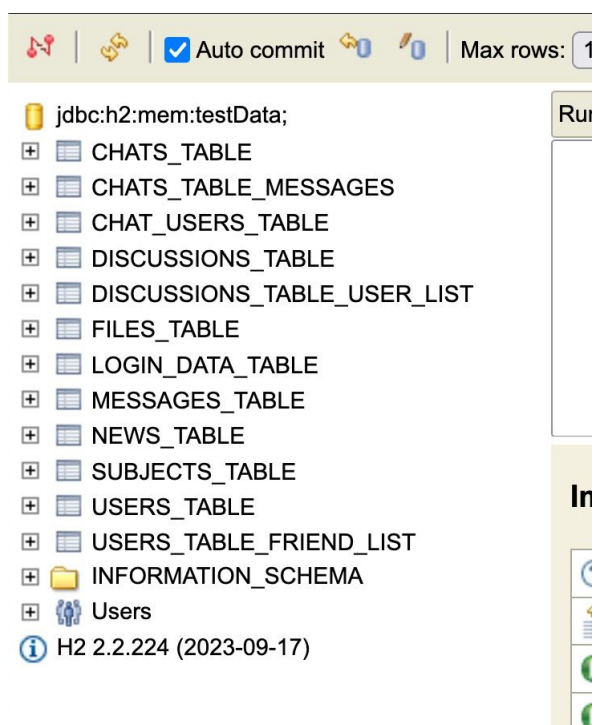


Рисунок 18 – Список таблиц базы данных сервиса

База данных сервиса строится на основе диаграммы классов, спроектированной в пункте 2.2 на рисунке 13. Для этого применяется технология Object-Relational Mapping (ORM).

ORM – это технология программирования, которая является посредником между написанным кодом приложения и базы данных и позволяет взаимодействовать с созданными в программе объектами в базе данных.

Запросы создания таблиц можно увидеть в листинге 1 и листинге 1 приложения В. Для создания таблиц применяется библиотека Hibernate, которая представляет возможности ORM.

Листинг 1 – Код создания таблиц базы данных

```
Hibernate: create table chat_users_table (chat_id bigint not null,
      user_id bigint not null)

Hibernate: create table chats_table (count_users_in_chat integer,
      private_chat boolean, chat_creator_id bigint,
      id bigint generated by default as identity,
      subject_id bigint, primary key (id))

Hibernate: create table chats_table_messages (chat_id bigint not null,
      messages_id bigint not null unique)

Hibernate: create table discussions_table (unread_message_count integer,
      discussion_chat_id bigint unique,
      id bigint generated by default as identity, owner_id bigint,
      subject_id bigint, description varchar(255) not null,
      term varchar(255) not null, title varchar(255) not null,
      primary key (id))

Hibernate: create table discussions_table_user_list
      (discussion_id bigint not null, user_list_id bigint not null)

Hibernate: create table files_table (created timestamp(6),
      file_owner_id bigint,
      id bigint generated by default as identity,
      file_name varchar(255), file_original_name varchar(255),
      file_type varchar(255), subject varchar(255),
      term varchar(255), file_content blob, primary key (id))

Hibernate: create table login_data_table
      (id bigint generated by default as identity,
      password varchar(30) not null,
      email varchar(255) unique, primary key (id))

Hibernate: create table messages_table
      (file_in_message_id bigint unique,
      id bigint generated by default as identity,
      message_author_id bigint, message_text varchar(255),
      primary key (id))

Hibernate: create table news_table (author_id bigint,
      id bigint generated by default as identity,
      content varchar(255),
      title varchar(255), primary key (id))
```

```

Hibernate: create table subjects_table
    (id bigint generated by default as identity,
     subject_name varchar(255), primary key (id))

Hibernate: create table users_table (friend_list_count integer,
    client_login_data_id bigint unique,
    id bigint generated by default as identity,
    role varchar(255), user_name varchar(255) unique,
    avatar blob, primary key (id))

Hibernate: create table users_table_friend_list
    (friend_list_id bigint not null,
     user_id bigint not null)

```

Всего в базе данных было создано 12 таблиц.

Между таблицами были настроены связи, показывающие их взаимодействие в базе. Код создания отношений представлен в листинге 2 и листинге 2 приложения В.

Листинг 2 – Код создания связей между таблицами

```

Hibernate: alter table if exists chat_users_table add constraint
    FKt7n9fry45x9ybudhe6wbxjib6 foreign key (user_id) references
    users_table

Hibernate: alter table if exists chat_users_table add constraint
    FKqo12kbxnh35wxy4k242ha6ukv foreign key (chat_id) references
    chats_table

Hibernate: alter table if exists chats_table add constraint
    FKiaiyurf5mbjsyplmqvt89371ca foreign key (chat_creator_id)
    references users_table

Hibernate: alter table if exists chats_table add constraint
    FK3tly70s6uscxxkg7x24tmov98 foreign key (subject_id)
    references subjects_table

Hibernate: alter table if exists chats_table_messages add constraint
    FKqy9e8oal7hx7i2elali4opig0 foreign key (messages_id)
    references messages_table

Hibernate: alter table if exists chats_table_messages add constraint
    FKjc73j989xtmd32hanvgqcxo7e foreign key (chat_id) references
    chats_table

Hibernate: alter table if exists discussions_table add constraint
    FKcyim7ya2w3a51pu3rr9h2no0v foreign key (discussion_chat_id)
    references chats_table

Hibernate: alter table if exists discussions_table add constraint
    FKmhusirfrmgst0se2v0bui9frk foreign key (owner_id) references
    users_table

Hibernate: alter table if exists discussions_table add constraint
    FKay7ffap41vr6bbmhv2bn7gs foreign key (subject_id)
    references subjects_table

Hibernate: alter table if exists discussions_table_user_list add
    constraint FK4jxdgrrybvdq9saypy0m2bvb foreign key
    (user_list_id) references users_table

Hibernate: alter table if exists discussions_table_user_list add
    constraint FK9dep3idsd2vq136k6amipvh2b foreign key

```

```

        (discussion_id) references discussions_table
Hibernate: alter table if exists files_table add constraint
FK5a00x3pcsmgngf4wuvq7lrcty foreign key (file_owner_id)
references users_table
Hibernate: alter table if exists messages_table add constraint
FKg4bl3yd12fdni9ndfxygouhci foreign key (message_author_id)
references users_table
Hibernate: alter table if exists messages_table add constraint
FKrbg5kuf61g8fgul5c760wbquq foreign key (file_in_message_id)
references files_table
Hibernate: alter table if exists news_table add constraint
FKsd0udb01l642vcbfwqmm6fofu foreign key (author_id) references
users_table
Hibernate: alter table if exists users_table add constraint
FKjf3ba8npny3kvbeutnnupxnk foreign key (client_login_data_id)
references login_data_table

```

После настройки таблиц и связей между ними их необходимо заполнить данными. Код заполнения таблиц приведён в листингах **3 – 14** приложения В.

Физическая схема разработанной базы данных сервиса представлена на рисунке 19.

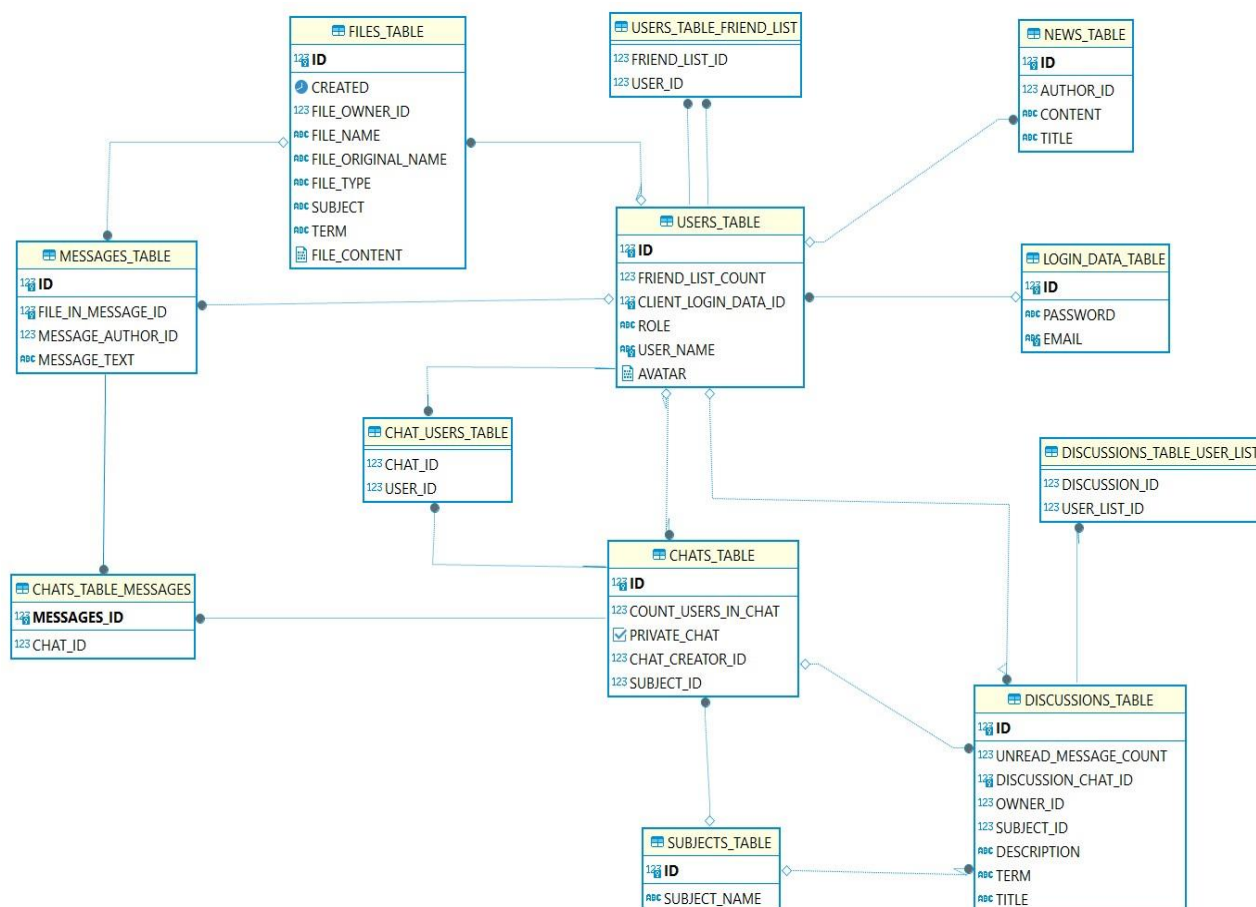


Рисунок 19 – Диаграмма базы данных сервиса для неформального взаимодействия участников образовательного процесса ВУЗа

Данная диаграмма наглядно отображает отношения между сущностями базы данных H2 и их атрибуты.

3.2 Разработка серверного программного обеспечения

Сервер в программном обеспечении – это участник клиент-серверной архитектуры, который содержит в себе основную логику приложения [22]. Он обрабатывает запросы от клиента, отправляет SQL-запросы к базе данных и наоборот получает их из БД. Основной код программы на сервере представлен в листингах приложения Г.

Точкой входа в программу является файл SocialNetworkApplication.java. После запуска программы для входа на страницу авторизации используется адрес: <http://localhost:8080/>. Все модели объектов базы данных описаны в пакете models. Краткое описание модулей представлено в таблице 9.

Таблица 9 – Описание модулей пакета models

Название модуля	Описание
Chat.java	Содержит описание полей и методов, хранящих данные о чатах
Discussion.java	Содержит описание полей и методов, хранящих данные об обсуждениях на Help-доске
File.java	Содержит описание полей и методов, хранящих данные о файлах
LoginData.java	Содержит описание полей и методов, хранящих данные о регистрациях пользователей
Message.java	Содержит описание полей и методов, хранящих данные о сообщениях пользователей
News.java	Содержит описание полей и методов, хранящих данные о новостях
Subject.java	Содержит описание полей и методов, хранящих данные о предметах
User.java	Содержит описание полей и методов, хранящих данные о пользователях

При выполнении запросов обращение к данным происходит в репозитории. Репозитории инкапсулируют логику, необходимую для доступа к источникам данных. То есть они являются посредником между базой данных и

классами, запрашивающими данные, предоставляя к ним доступ через общие функции. В приложении реализовано 8 репозитория (интерфейсов):

- ChatRepository;
- DiscussionRepository;
- FileRepository;
- LoginDataRepository;
- MessageRepository;
- NewsRepository;
- SubjectRepository;
- UserRepository.

Для обработки HTTP-запросов от клиента используется контроллер. Он определяет какие данные нужно обрабатывать и какой результат возвращать клиенту. Всего в программе определено 18 контроллеров. Краткое описание наиболее значимых по функционалу контроллеров приведено в таблице 10.

Таблица 10 – Описание модулей пакета controller

Название модуля	Описание
AddUserToDiscussionController.java	Обрабатывает запросы по добавлению пользователя или группы пользователей в обсуждение
ChatPageController.java	Обрабатывает запросы, связанные с действиями в чате
CreateAccountController.java	Обрабатывает запросы по созданию аккаунта пользователя
CreateDiscussionController.java	Обрабатывает запросы по созданию обсуждений, размещенных на Help-доске
DiscussionPageController.java	Обрабатывает запросы по открытию и размещению сообщений в обсуждении на Help-доске
HelpDeskController.java	Обрабатывает запросы, связанные непосредственно с Help-доской и её открытием
UserLibraryController.java	Обрабатывает запросы, связанные с нахождением файлов по критериям и наименованию

Веб-приложение было разработано с помощью использования фреймворка Spring Boot. Он упрощает создание автономных приложений производственного

уровня, которые можно легко развернуть и запустить с минимальной конфигурацией.

3.3 Разработка клиентского программного обеспечения

Клиент в программном обеспечении – это участник клиент-серверной архитектуры, с которым непосредственно работает пользователь через веб-браузер. Он даёт возможность пользователю через него просматривать информацию, редактировать её и отправлять серверу для обработки.

Клиент сервиса состоит из нескольких связанных между собой страниц, созданных с помощью языка разметки HTML. Их код представлен в листингах приложения Д.

Взаимодействие между страницами сервиса для неформального взаимодействия участников образовательного процесса ВУЗа показано на диаграмме компонентов, представленной на рисунке 20.

На диаграмме отображены следующие компоненты:

- `SocialNetworkApplication.java` – точка запуска приложения;
- `SocialNetworkApplication.class` – класс, из которого идёт обращение к формам;
- `login_page.html` – форма авторизации пользователя;
- `create_account_page.html` – форма регистрации пользователя;
- `admin_page.html` – страница администратора, на которой видны все пользователи с ролями;
- `file_lib.html` – страница, содержащая личные файлы;
- `friends_page.html` – страница, на которой отображены друзья пользователя;
- `friend_page.html` – личная страница конкретного друга;
- `help_desk_page.html` – страница, на которой находится Help-доска;
- `create_discussion_form.html` – форма создания обсуждения на Help-доске;
- `add_user_discussion.html` – форма, на которой определяется круг пользователей-получателей доступ к обсуждению;

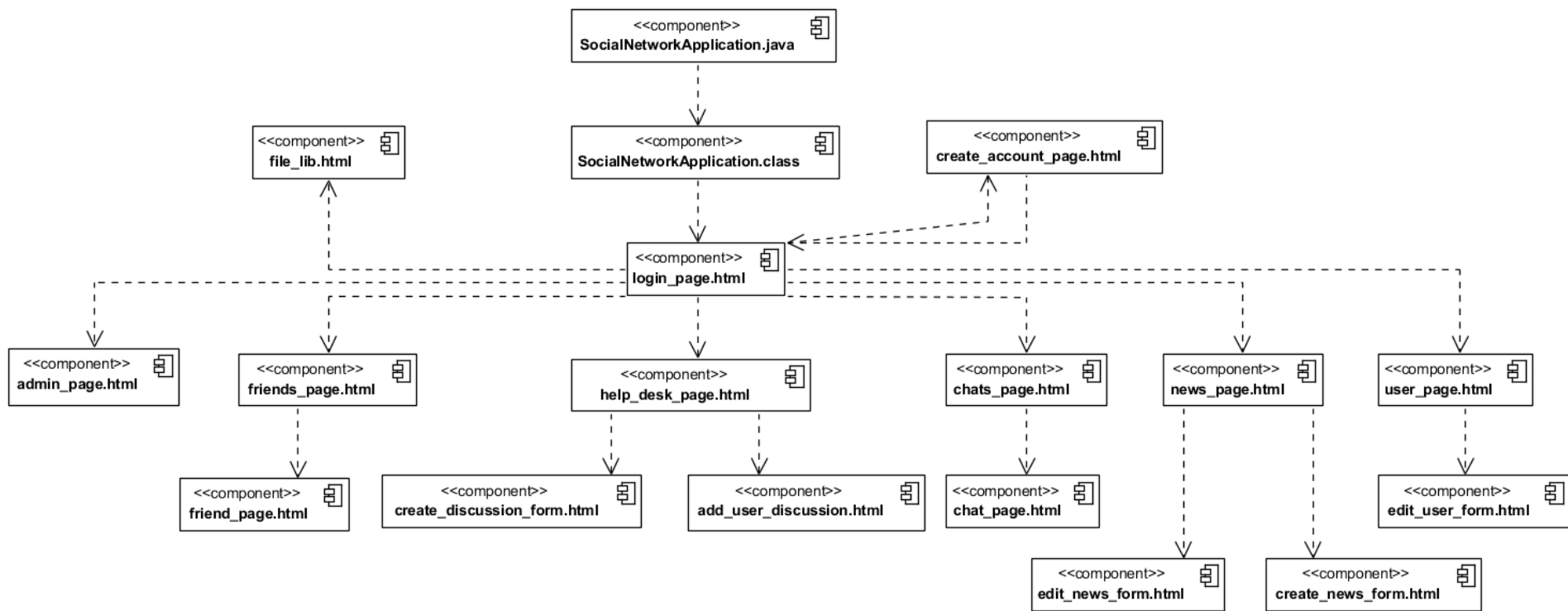


Рисунок 20 – Диаграмма компонентов клиентской части сервиса

- chats_page.html – страница со списком всех чатов пользователя;
- chat_page.html – конкретный чат пользователя;
- news_page.html – страница со всеми новостями;
- create_news_form.html – форма создания новости пользователя;
- edit_news_form.html – форма редактирования новости пользователя;
- user_page.html – личный профиль, предоставляющий информацию о пользователе;
- edit_user_form.html – форма редактирования личной информации пользователя.

На основе разработки клиентской части сервиса был получен пользовательский интерфейс, реализующий большую часть функционала, описанного в задачах по разработке сервиса. На рисунке 21 представлен интерфейс раздела «Моя библиотека», в котором пользователь хранит личные файлы и может искать материалы других пользователей. Также он может производить поиск документов по курсу, предмету, наименованию или дате.

Сверху расположена панель управления, на которой есть переход в разделы «Новости», «Help-доска», «Кабинет пользователя», «Друзья», «Чаты», «Библиотека файлов» и «Выход».

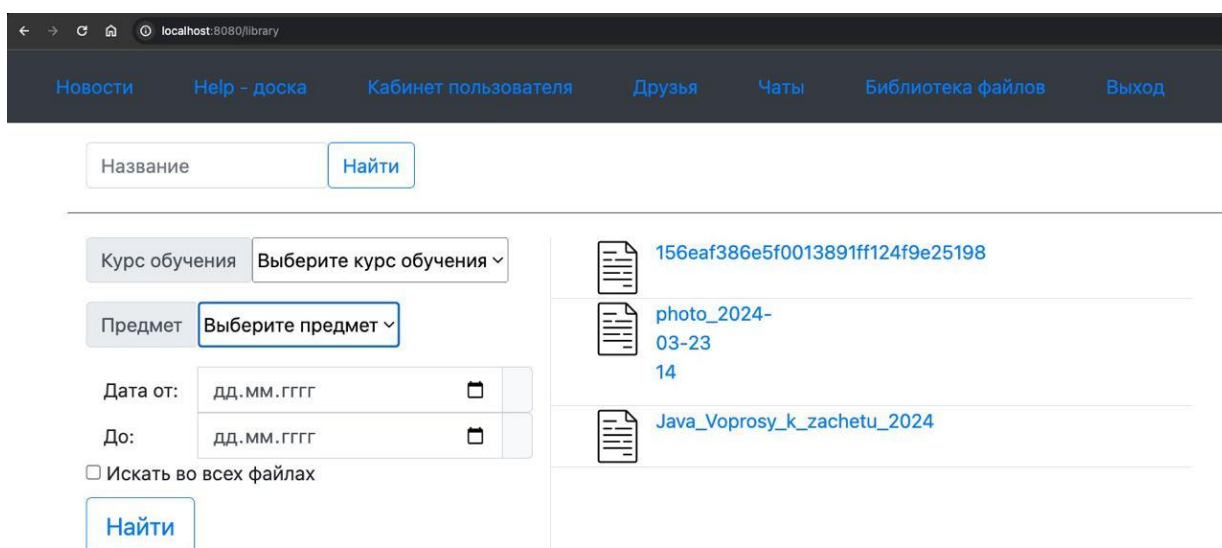
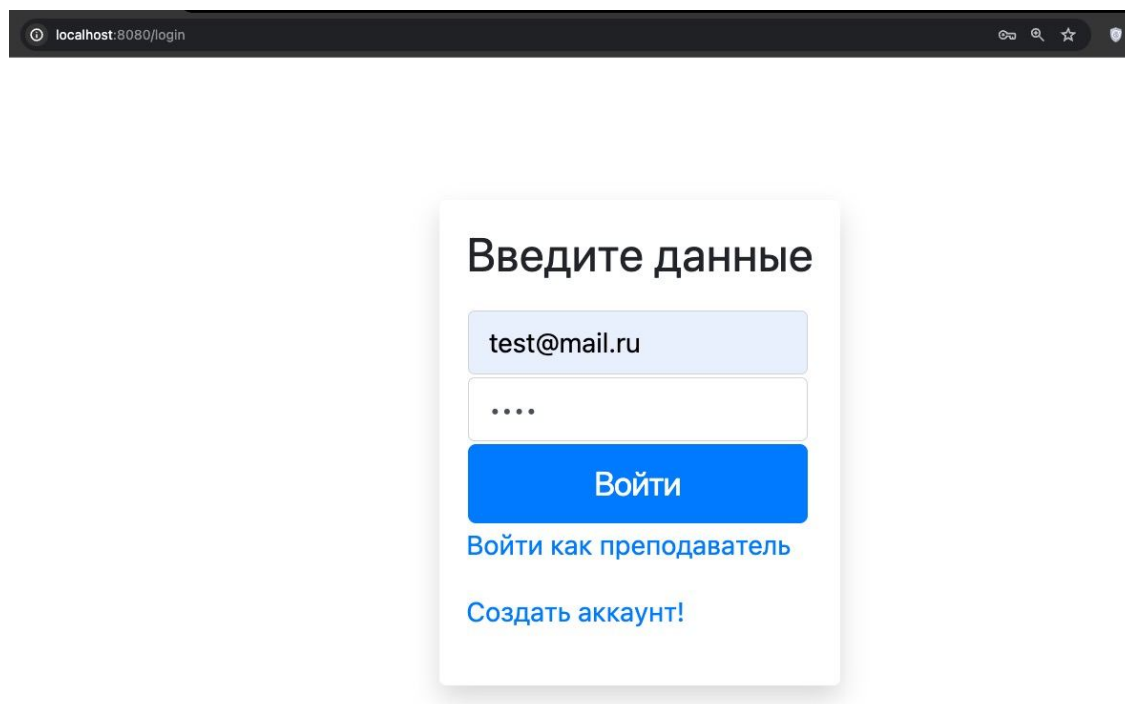


Рисунок 21 – Интерфейс раздела «Библиотека файлов»

Переход к сервису осуществляется со страницы авторизации, представленной на рисунке 22.



The image shows a web browser window with the address bar displaying 'localhost:8080/login'. The main content is a login form with the following elements:

- Title: **Введите данные**
- Email input field: `test@mail.ru`
- Password input field: `....`
- Blue button: **Войти**
- Link: [Войти как преподаватель](#)
- Link: [Создать аккаунт!](#)

Рисунок 22 – Интерфейс страницы авторизации пользователя на сервисе

На странице авторизации предусмотрена возможность входа как студента, так и преподавателя по отдельной ссылке. Если пользователь не зарегистрирован, он может пройти регистрацию, нажав по ссылке «Создать аккаунт!».

4 Контроль качества программного обеспечения сервиса для неформального взаимодействия участников образовательного процесса ВУЗа

4.1 Планирование процесса контроля качества программного обеспечения

4.1.1 Обоснование выбора модели тестирования

Тестирование программного обеспечения (ПО) можно условно поделить на 3 вида:

- тестирование структуры;
- функциональное тестирование;
- нефункциональное тестирование.

Тестирование структуры данных – это вид тестирования ПО, в котором в качестве объекта тестирования является внутренний дизайн программного обеспечения. Структурное тестирование ориентировано на обнаружение ошибок внутри кода программы.

Функциональное тестирование – один из ключевых типов тестирования программного обеспечения. Главной задачей функционального тестирования является проверка соответствия разработанного ПО исходным функциональным требованиям. Наиболее распространенными типами функционального тестирования являются тестирование белого ящика и черного ящика.

Тестирование методом «черного ящика» – это метод проверки программного обеспечения, при котором тестировщик не знает его внутреннюю структуру. Он не имеет доступа к реализации приложения, поэтому тестирование происходит на основе анализа поведения системы. Это позволяет определять реакцию системы на ожидаемые и неожиданные действия пользователя, время реакции и проблемы с надёжностью.

Проверяя методом черного ящика, тестировщик начинает с изучения спецификации тестирования программного обеспечения. После чего он вводит различные входные данные на основе заранее заданного набора тест-кейсов.

После проведения тестирования он просто сообщает разработчикам о выявленных ими проблемах, не анализируя их возникновение.

При тестировании белым ящиком разработчик анализирует внутреннюю структуру программы. Он знает, как устроен код программного обеспечения и какие функции он выполняет в конкретном месте. То есть основной объект тестирования таким способом – это код системы. При тестировании разработчик изучает структуру каталогов, циклы, процедуры и самостоятельно может вносить в них изменения.

Тестирование белого ящика можно разделить на две стадии:

- исследование программного кода;
- разработка тест-кейсов и других алгоритмов проверки кода.

Нефункциональное тестирование включает в себя такие виды тестирования, как стресс-тестирование, нагрузки, минимальные системные требования, тестирование рисков в базе данных и проверка работы с производительностью базы данных.

Тестирование пользовательского интерфейса (User Interface testing) – один из видов тестирования программного обеспечения, проверяющий корректность работы и удобство использования пользовательского интерфейса приложения. Данный вид тестирования является одним из наиболее важных, потому что качество пользовательского интерфейса напрямую влияет на готовность пользователей продолжать использовать приложение.

Тестирование пользовательского интерфейса состоит из проверки следующих аспектов:

- правильность отображения на различных устройствах и при разном разрешении, соответствие дизайна макетам, соблюдение цветовой гаммы и стилей;
- проверка правильной работы кнопок, полей ввода, выбора и других элементов управления, а также их доступности для пользователя;
- проверка удобства навигации, перемещения между разделами приложения.

Для тестирования приложения в качестве функционального вида тестирования было выбрано тестирование методом «белого ящика», исходя из возможности знания внутренней структуры программы и соответственно более быстрого понимания и решения возможных проблем.

4.1.2 Обоснование выбора метрик программного кода

Метрика тестирования программного обеспечения – это стандарт или критерий, используемый для отслеживания эффективности усилий, направленных на обеспечение качества программного продукта. Он представляет собой набор оценок системы, которые, в свою очередь, дают более глубокое представление о коде.

Метрики делятся на следующие категории:

- количественные метрики;
- метрики сложности потока управления программы;
- метрики сложности потока управления данными.

К количественным метрикам относится Source Lines of Code (SLOC) – метрика подсчёта количества строк кода, которая первоначально была разработана для оценки затрат по проекту. Однако с появлением языков программирования, позволяющих записывать несколько команд в одну строку, метрика стала практически неприменимой. Поэтому код был разделен на физические и логические строки, в которых логические строки – это количество команд системы, а физические – общее количество строк всей программы.

Из-за недостатков применения данной метрики, связанных с зависимостью от используемого языка программирования и его стиля, для оценки качества кода в совокупности используют метрики Холстеда и метрики Джиба. Они позволяют учесть насыщенность условными операторами и циклами и на их основе определить сложность программного обеспечения.

Метрики сложности потока управления программы основаны на анализе управляющего графа. Наиболее распространённой оценкой данного класса является цикломатическая сложность программы или цикломатическое число

Мак-Кейба. Она основана на анализе получившегося графа, а именно: на количестве дуг, вершин, компонент связности. Однако данная метрика не способна различать циклические и условные конструкции.

Для компенсации этого недостатка были разработаны метрики Г. Майерса и метод Хансена. Метрика Г. Майерса позволяет различать разные по сложности предикаты, однако на практике он почти не применяется. В соответствии с методом Хансена мера сложности потока управления программы определяется в виде пары цикломатической сложности и числа операторов. Данный метод позволяет различать разные по сложности предикаты, однако на практике он почти не применяется. Преимуществом данной меры является ее чувствительность к структурированности ПО.

К метрикам сложности потока управления данными относятся метрика Чепина и метрика Спена.

Суть метода Чепина заключается в анализе характера использования переменных программного кода. Данная оценка позволяет определить информационную прочность в данном программном модуле. Для этого все переменные делятся на 4 функциональные группы: переменные, вводимые для расчетов и для обеспечения вывода (Р); создаваемые внутри программы (М); переменные, участвующие в управлении работой модуля (С); не используемые в программе переменные (Т) [23].

Метрика Спена основывается на том, какие данные используются внутри каждой программной секции. Спен – это число утверждений, содержащих данный идентификатор, между его первым и последним появлением в тексте программы. Следовательно, идентификатор, появившийся n раз, имеет спен, равный $n-1$. Большое число утверждений затрудняет тестирование и отладку программы.

Для тестирования программного кода были выбраны следующие классы: метрика SLOC, метрика Холстеда и метрика Чепина.

4.2 Тестирование программных средств

4.2.1 Тестирование белым ящиком

Тестирование белым ящиком – это вид функционального тестирования, при котором известна внутренняя структура программного кода. Для выполнения тестирования были взяты функции `createAccount()`, в которой происходит создание аккаунта пользователя; `updateUser()` для редактирования данных пользователя; `sendMessageToDiscussion()`, реализующая операцию отправки сообщения в обсуждение; `addUserToDiscussion()`, необходимая для добавления пользователей в обсуждение. В качестве примера выполнения тестирования белым ящиком предложен метод `createAccount()` (листинг 3).

Листинг 3 – Метод `createAccount()`

```
@PostMapping("/createAccount/new")
public String createAccount(@Valid LoginData loginData,
                           @RequestParam(name = "clientName")
                           String clientName, Errors errors, Model
                           model)
{
    if (errors.hasErrors()) {
        model.addAttribute("message", "Неверный формат данных
                               клиента!");
        return "create_account_page";
    }
    if (userService.createUser(loginData.getEmail(),
                               loginData.getPassword(), clientName, "user").getId()
        != 0) {
        model.addAttribute("message", "Пользователь добавлен!");
        return "create_account_page";
    }
    model.addAttribute("message", "Ошибка добавления
                           пользователя!");
    return "create_account_page";
}
```

В результате анализа данного кода была построена блок-схема, представленная на рисунке 23. Она состоит из двух условий, первое из которых определяет, была ли ошибка при записи данных пользователя. Если да, то программа выводит ошибку «Неверный формат данных клиента!». Второе условие определяет, добавлен такой пользователь или ещё нет. Если данные пользователя были получены, то выдаётся сообщение о том, что пользователь добавлен. Иначе происходит ошибка добавления пользователя.

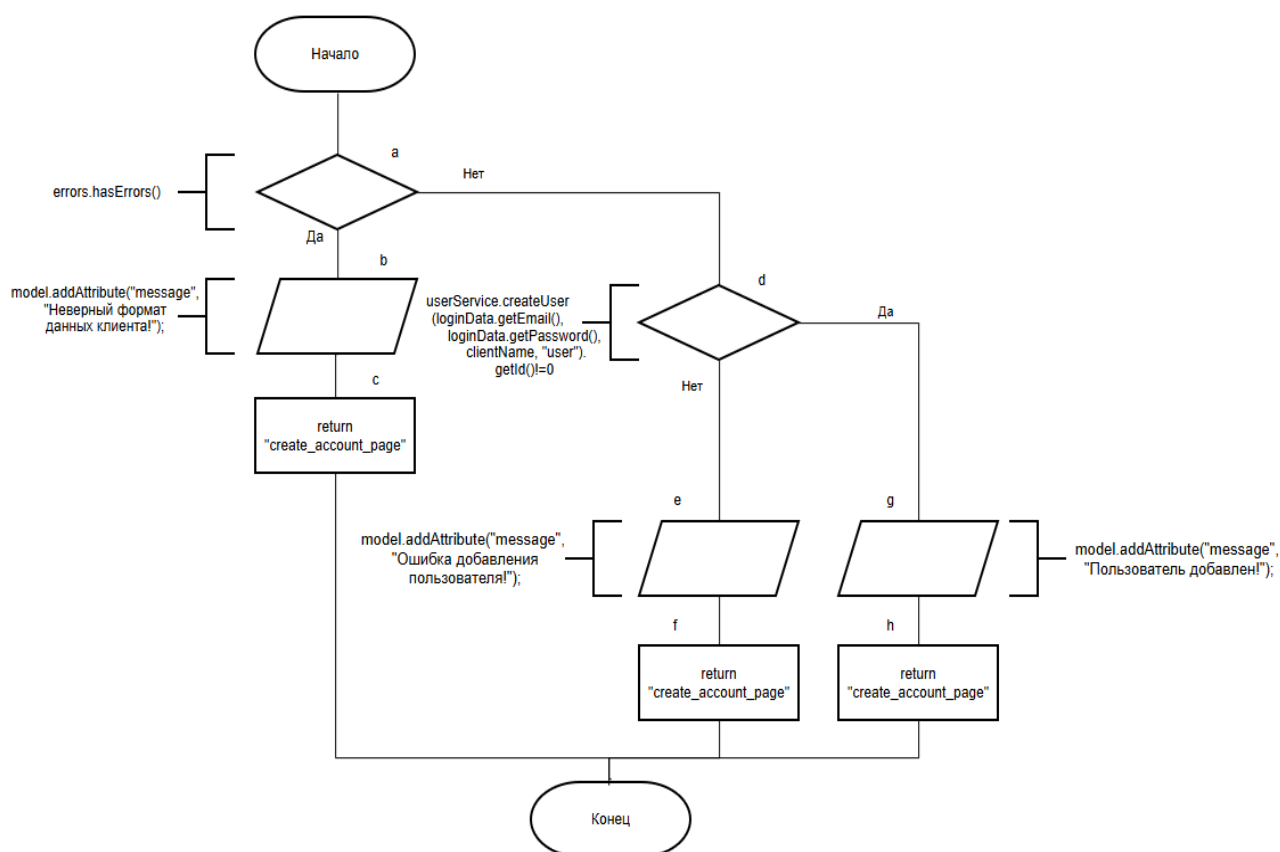


Рисунок 23 – Блок-схема метода createAccount()

В качестве критерия тестирования выбрано покрытие решений и условий. В соответствии с этим критерием необходимо составить тесты так, чтобы результаты каждого условия выполнялись хотя бы один раз, результаты каждого решения так же выполнялись хотя бы один раз, и каждый оператор должен быть выполнен хотя бы один раз. Содержимое и результаты тестирования покрытием решений и условий представлены в таблице 11.

Таблица 11 – Покрытие решений и условий

№	Условие	Входные данные	Выходные данные
1	a(да), b, c	errors.hasErrors = true	Сообщение: «Неверный формат данных клиента!»
2	a(нет), d(нет), e, f	errors.hasErrors = false, userService.createUser = 0	Сообщение: «Ошибка добавления пользователя!»
3	a(нет), d(да), g, h	errors.hasErrors = false, userService.createUser = 2	Сообщение: «Пользователь добавлен!»

В ходе выполнения тестирования белым ящиком были получены успешные результаты, а, следовательно, данная часть кода работает корректно.

4.3 Оценка качества программного кода на основе метрик

4.3.1 Метрика SLOC

Количество строк кода (англ. Source Lines of Code – SLOC) – это метрика программного обеспечения, используемая для измерения его объёма с помощью подсчёта количества строк в тексте исходного кода. Проведем подсчет данной метрики в функции `findByCriteria`, служащей для нахождения файлов в личной библиотеке по различным критериям отбора. Функция приведена в листинге 4.

Листинг 4 – Метод `findByCriteria`

```
//Метод отбора файлов
@PostMapping("/files/findByCriteria")
public String findByCriteria(@RequestParam String term,
                             @RequestParam String subj,
                             @RequestParam String dateFrom,
                             @RequestParam String dateTo,
                             Model model, HttpSession session)
    throws ParseException {
    boolean termExist = term != null && !term.isBlank();
    boolean subjExist = subj != null && !subj.isBlank();
    boolean dateExist = dateFrom != null &&
        !dateFrom.isBlank() &&
        dateTo != null && !dateTo.isBlank();
    Date from = new Date();
    Date to = new Date();

    if (dateExist) {
        SimpleDateFormat formater = new SimpleDateFormat("yyyy-MM-");
        from = formater.parse(dateFrom);
        to = formater.parse(dateTo);
    }

    List<File> files = new ArrayList<>();

    if (termExist && subjExist && dateExist) {
        files = fileRepository.
            findAllByTermAndSubjectAndCreatedBetween(term, subj,
            from, to);
    } else if (termExist && subjExist) {
        files = fileRepository.findAllByTermAndSubject(term, subj);
    } else if (termExist && dateExist) {
        files = fileRepository.findAllByTermAndCreatedBetween
            (term, from, to);
    } else if (subjExist && dateExist) {
        files = fileRepository.
            findAllBySubjectAndCreatedBetween(subj, from, to);
    } else if (termExist) {
        files = fileRepository.findAllByTerm(term);
    } else if (subjExist) {
        files = fileRepository.findAllBySubject(subj);
    }
}
```

```

    } else if(dateExist) {
        files = fileRepository.findAllByCreatedBetween(from, to);
    }
    model.addAttribute("files", files);
    User user = (User) session.getAttribute("currentUser");
    user = userRepository.getReferenceById(user.getId());
    model.addAttribute("currentUser", user);
    return "file_lib";
}

```

Физическими строками считаются все непустые строки текстового файла.

Измеряя логические строки кода, предпринимается попытка посчитать количество собственно операторов в программе, но, разумеется, их определение зависит от конкретного языка программирования.

В данной программе получаются следующие значения:

- 45 физических строк кода;
- 42 логических строк кода;
- 1 строка комментариев.

Значение метрики SLOC равняется 45 физическим строкам кода и 42 логическим. Это нормальные значения для одной функции.

4.3.2 Метрика Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

- NUOprtr (Number of Unique Operators) – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
- NUOprnd (Number of Unique Operands) – число уникальных операндов программы (словарь операндов);
- Noprtr (Number of Operators) – общее число операторов в программе;
- Noprnd (Number of Operands) – общее число операндов в программе.

Операторы и операнды программного кода, представленного в листинге 4, приведены в таблицах 12 и 13 соответственно.

Таблица 12 – Операторы метода findByCriteria

Оператор	Число вхождений
=	18
!=	4
!	4
if	2
else if	6
&&	10
@	4
;	20
.	16
,	16
{}	9
return	1
throws	1
.isBlank()	4
null	4
new	4
Date()	2
.SimpleDateFormat()	1
.parse	2
ArrayList<>()	1
.findAllByTermAndSubjectAndCreatedBetween()	1
.findAllByTermAndSubject()	1
.findAllBySubjectAndCreatedBetween()	1
.findAllByTerm()	1
.findAllBySubject()	1
.findAllByCreatedBetween()	1
.addAttribute()	2
.getAttribute()	1
.getReferenceById()	1
n1 = 30	N1 = 139

Таблица 13 – Операнды метода findByCriteria

Операнд	Число вхождений
«/files/findByCriteria»	1
term	7
subj	6
dateFrom	2
dateTo	2
model	3
session	2
termExist	5
subjExist	5
dateExist	6
from	6
to	6
files	9

Продолжение таблицы 13

Операнд	Число вхождений
formater	3
fileRepository	7
user	4
«files»	1
«currentUser»	1
«file_lib»	1
n2 = 20	N2 = 77

Длина программы (N) рассчитывается по формуле:

$$N = N1 + N2.$$

По результатам расчёта длина программы равна 216. Словарь программы (n) рассчитывается по формуле:

$$n = n1 + n2.$$

Словарь программы равен 50. Объем программы (HV) рассчитывается по формуле:

$$HV = N \times \log_2(n).$$

Объём программы в ходе умножения длины программы на логарифм словаря программы оказался равен 1219,104. Сложность программы (HD) рассчитывается по формуле:

$$HD = \frac{n1}{2} \times \frac{N2}{n2}.$$

По результатам расчёта с помощью использования количества операторов и операндов, а также числа вхождений операндов равен 57,75.

4.3.3 Метрика Чепина

Суть метода состоит в оценке информационной прочности отдельно взятого программного модуля с помощью анализа характера использования переменных из списка ввода-вывода.

Все множество переменных, составляющих список ввода-вывода, разбивается на 4 функциональные группы:

- 1) вводимые переменные для расчетов и для обеспечения вывода (P);

- 2) модифицируемые, или создаваемые внутри программы переменные (M);
- 3) переменные, участвующие в управлении работой программного модуля, управляющие переменные (C);
- 4) не используемые в программе («паразитные») переменные (T).

Поскольку каждая переменная может выполнять одновременно несколько функций, необходимо учитывать ее в каждой соответствующей функциональной группе. Формула метрика Чепина следующая:

$$Q = a1 \times P + a2 \times M + a3 \times C + a4 \times T,$$

где $a1$, $a2$, $a3$, $a4$ – весовые коэффициенты.

Весовые коэффициенты использованы для отражения различного влияния на сложность программы каждой функциональной группы. По мнению автора метрики, наибольший вес, равный 3, имеет функциональная группа C, так как она влияет на поток управления программы. Весовые коэффициенты остальных групп распределяются следующим образом: $a1=1$, $a2=2$, $a4=0.5$. Весовой коэффициент группы T не равен 0, поскольку «паразитные» переменные не увеличивают сложность потока данных программы, но иногда затрудняют ее понимание. С учетом весовых коэффициентов получается следующая формула:

$$Q = P + 2M + 3C + 0.5T.$$

На основе тестирования программного кода из листинга 4 получены следующие значения коэффициентов:

- 1) P равно 7;
- 2) M равно 8;
- 3) C равно 5;
- 4) T равно 0.

Значение метрики Чепина равняется 38. На основе полученных значений метрики Чепина уровень сложности данного решения можно считать сравнительно нормальным, как так в исходном тексте программы используется достаточное количество переменных, что не затрудняет понимание программы.

В ходе тестирования сервиса для неформального взаимодействия участников процесса ВУЗа было выполнено функциональное и нефункциональное тестирование. Результаты различных видов тестирования показали, что программное обеспечение работает исправно и надёжно. Оценка качества программного кода на основе метрик показала эффективную работу сервиса. Из всего вышесказанного следует, что контроль качества программного обеспечения прошёл успешно.

Заключение

В ходе выполнения выпускной квалификационной работы была достигнута поставленная цель разработки программного сервиса для неформального взаимодействия участников образовательного процесса ВУЗа. Для её достижения были решены следующие задачи:

- выполнен анализ предметной области и постановка задачи на разработку сервиса;
- выполнено планирование процесса разработки сервиса;
- выполнение проектирование программного сервиса;
- разработано программное обеспечение (база данных, клиент-серверная часть приложения);
- выполнено тестирование программного сервиса.

На основании результатов контроля качества программного обеспечения можно сделать вывод о том, что сервис для неформального взаимодействия участников образовательного процесса ВУЗа работает качественно и без значительных сбоев.

Список использованных источников

1. Ковалева А.И. Образовательный процесс в вузе и отражение его хода в электронной информационной образовательной среде // cyberleninka.ru. Теория. [Электронный ресурс] URL: <https://cyberleninka.ru/article/n/obrazovatelnyy-protsess-v-vuze-i-otrazhenie-ego-hoda-v-elektronnoy-informatsionnoy-obrazovatelnoy-srede/viewer> (дата обращения: 12.04.2024).
2. Права и обязанности участников образовательного процесса // Уполномоченный по правам ребёнка в Калужской области. [Электронный ресурс] URL: <https://kalugapravo.ru/vazhno-dlya-vzroslyh/obrazovanie/prava-i-obyazannosti-uchastnikov-obrazovatel'nogo-processa?ysclid=lvb79jd7n5915499171> (дата обращения: 12.04.2024).
3. Федеральный закон от 22.08.1996 N 125-ФЗ (ред. от 03.12.2011) «О высшем и послевузовском профессиональном образовании» (с изм. и доп., вступающими в силу с 01.02.2012) // КонсультантПлюс. [Электронный ресурс] URL: https://www.consultant.ru/document/cons_doc_LAW_11446/3e4dd661d759b03776b63710d5b0ce3b21bd2364/?ysclid=lvb7tq0rzl818466241 (дата обращения: 12.04.2024).
4. Формальные и неформальные взаимоотношения // dzen.ru. Теория. [Электронный ресурс] URL: <https://dzen.ru/a/Xxv42mA0TyOxSpCd> (дата обращения: 13.04.2024).
5. Филонов Н. Campus. Система, позволяющая управлять образовательным процессом // habr.com. Теория. [Электронный ресурс] URL: <https://habr.com/ru/articles/719596/> (дата обращения: 15.04.2024).
6. Рудов К., Коломак Т. Тред // Unisender. Словарь маркетолога. [Электронный ресурс] URL: <https://www.unisender.com/ru/glossary/chto-takoe-tred-i-zachem-on-nuzhen/> (дата обращения: 15.04.2024).
7. Диаграмма деятельности // ru.wikipedia.org. Теория. [Электронный ресурс] URL:

https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_%D0%B4%D0%B5%D1%8F%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8 (дата обращения: 18.04.2024).

8. Бударин М. Как организовать процесс программной разработки // simpleone.ru. [Электронный ресурс] URL: <https://simpleone.ru/blog/proczess-razrabotki-programmnogo-obespecheniya/?ysclid=lw3d9r6l9d739013395> (дата обращения: 19.04.2024).

9. Чепурнова Д. Разбираем диаграмму Ганта – инструмент, который должен знать каждый менеджер // Skillbox Media. Управление. [Электронный ресурс] URL: <https://skillbox.ru/media/management/razbiraem-diagrammu-ganta-instrument-kotoryy-dolzhen-znat-kazhdyy-menedzher/?ysclid=lw3d359259905170168> (дата обращения: 19.04.2024).

10. Воронова М., Павлова А. Диаграммы последовательности – простой способ управления процессами для аналитиков // practicum.yandex.ru. Анализ данных. [Электронный ресурс] URL: <https://practicum.yandex.ru/blog/sequence-diagram/> (дата обращения: 28.04.2024).

11. Фаулер, М. UML. Основы / 3 е издание. – Пер. с англ. – СПб: Символ Плюс, 2004 – 192 с.

12. Case-средства проектирования баз данных // hsbi.hse.ru. Статьи. [Электронный ресурс] URL: <https://hsbi.hse.ru/articles/case-sredstva-proektirovaniya-baz-dannykh/?ysclid=lwfcibi13s37616152> (дата обращения: 01.05.2024).

13. Серебрякова Т.А., Тюриков И.А. Сравнительный анализ case-средств // cyberleninka.ru. Теория. [Электронный ресурс] URL: <https://cyberleninka.ru/article/n/sravnitelnyi-analiz-case-sredstv?ysclid=lwfd9mpuom650206048> (дата обращения: 03.05.2024).

14. Атаманов, Ю. С. Обзор базовых возможностей ERwin Data Modeler / Ю. С. Атаманов, В. С. Гончарук, С. Н. Гордеев. – Текст: непосредственный //

Молодой ученый. – 2017. – № 9 (143). – С. 26-28. [Электронный ресурс] URL: <https://moluch.ru/archive/143/40104/> (дата обращения: 12.05.2024).

15. Ерёмин, А. Д. Методология науки / Ерёмин А.Д. – 3-е изд.. – Саров: СарФТИ, 2019 – 36 с.

16. Искусство кода. 10 популярных языков программирования // blog.skillbox.ru. Код. [Электронный ресурс] URL: <https://blog.skillbox.by/kod/iskusstvo-koda-10-populjarnyh-jazykov-programmirovanija/> (дата обращения: 14.05.2024).

17. Java // blog.skillfactory.ru. Глоссарий. [Электронный ресурс] URL: <https://blog.skillfactory.ru/glossary/java/> (дата обращения: 19.05.2024).

18. SQL (язык программирования) // bigenc.ru. Языки программирования. [Электронный ресурс] URL: <https://bigenc.ru/c/sql-iazyk-programmirovaniia-47e1bd> (дата обращения: 20.05.2024).

19. Perfomance // h2database.com. [Электронный ресурс] URL: <https://h2database.com/html/performance.html> (дата обращения: 20.05.2024).

20. Редакция сайта Geekbrains Пользовательский интерфейс: виды и правила создания // gb.ru. Блог. [Электронный ресурс] URL: <https://gb.ru/blog/polzovatelskij-interfejs/> (дата обращения: 21.05.2024).

21. Бирман, И. Б. Пользовательский интерфейс / И. Б. Бирман. — М. : Изд-во бюро Горбунова, 2017. – 63 с.

22. Что такое Веб-приложение // vsmi.github.io. [Электронный ресурс] URL: <https://vsmi.github.io/testing-po/testing-part-6/> (дата обращения: 25.05.2024).

23. Рыжков Е. Программный код и его метрики // habr.com. Теория. [Электронный ресурс] URL: <https://habr.com/ru/companies/intel/articles/106082/> (дата обращения: 28.05.2024).

Приложение А
(справочное)
Глоссарий

Студент – лицо, в установленном порядке зачисленное в высшее учебное заведение для обучения.

Педагогический работник – лицо, профессионально занимающееся образовательной деятельностью. В состав педагогических работников входят преподаватели, методисты, административно-педагогический персонал (деканаты, кафедры и т. д.) и другие участники.

Социальная сеть – это приложение, предназначенное для неформального общения пользователей вне зависимости от статуса или роли, обмена разного рода информацией, создания отношений между людьми, имеющими схожие интересы.

Электронная информационно-образовательная среда (по др. ЭИОС) – система, которая объединяет различные информационные ресурсы и инструменты для удобного взаимодействия студентов, преподавателей и сотрудников университета.

Интернет-платформа для дополнительной помощи при обучении – это информационная система, предоставляющая дополнительные возможности в образовательном процессе.

Диаграмма Ганта – это визуальное представление графика работ, построенное согласно плану проекта.

Прецеденты – это технология определения функциональных требований к системе.

Актер (actor) – роль, которую пользователь играет по отношению к системе.

Спецификация – это описание последовательности действий пользователя в системе в рамках одного варианта использования.

Концептуальная модель – модель, представленная множеством понятий и связей между ними, определяющих смысловую структуру рассматриваемой предметной области или её конкретного объекта.

Архитектура клиент-сервер – это схема организации вычислительных систем, в которой выполнение задач распределено между клиентом и сервером с помощью сетевого взаимодействия.

Диаграмма последовательности – это наглядное представление совокупности разных элементов модели системы, изображение того, как и в каком порядке они взаимодействуют.

Пользовательский интерфейс UI (User Interface) – это набор правил и материальные средства, которые обеспечивают взаимодействие системы и пользователя по этим правилам.

CASE-средства (Computer - Aided Software Engineering) – это методы и технологии, которые позволяют проектировать различные информационные системы (в частности, базы данных) и автоматизировать их создание.

Visual Paradigm – это инструмент, используемый для создания и управления версиями UML или унифицированного языка моделирования, который также поддерживает группу управления объектами, включая нотацию для моделирования бизнес-процессов.

Integrated Development Environment (IDE) – это интегрированная среда разработки, используемая разработчиками для создания ПО.

Java – это объектно-ориентированный язык программирования, который появился в мае 1995 года благодаря компании Sun Microsystems.

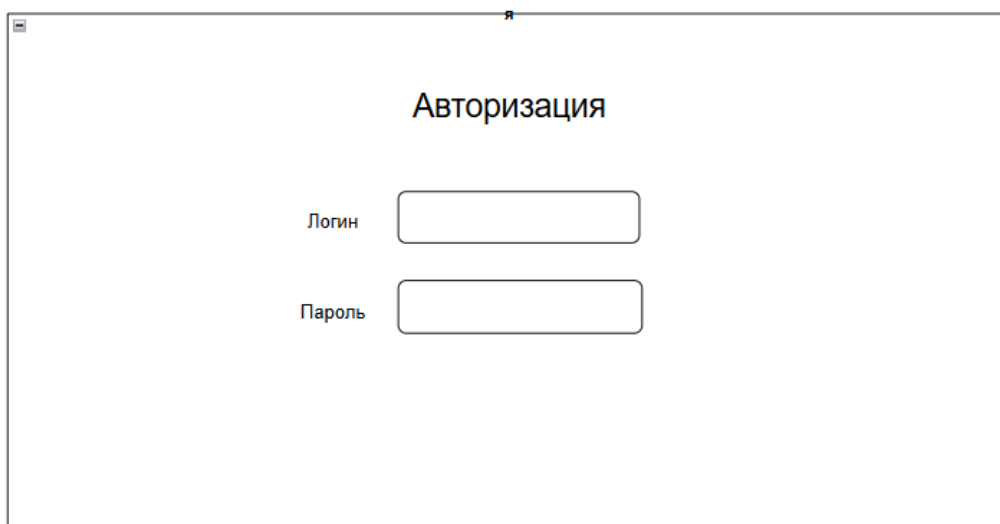
H2 – открытая кроссплатформенная система управления базами данных, полностью написанная на языке Java.

База данных – это упорядоченный определённым образом набор структурированных данных, которые чаще всего сохраняются в электронном виде в компьютерной системе

Сервер в программном обеспечении – это участник клиент-серверной архитектуры, который содержит в себе основную логику приложения.

Клиент в программном обеспечении – это участник клиент-серверной архитектуры, с которым непосредственно работает пользователь через веб-браузер.

Приложение Б
(рекомендуемое)
Шаблоны пользовательского интерфейса

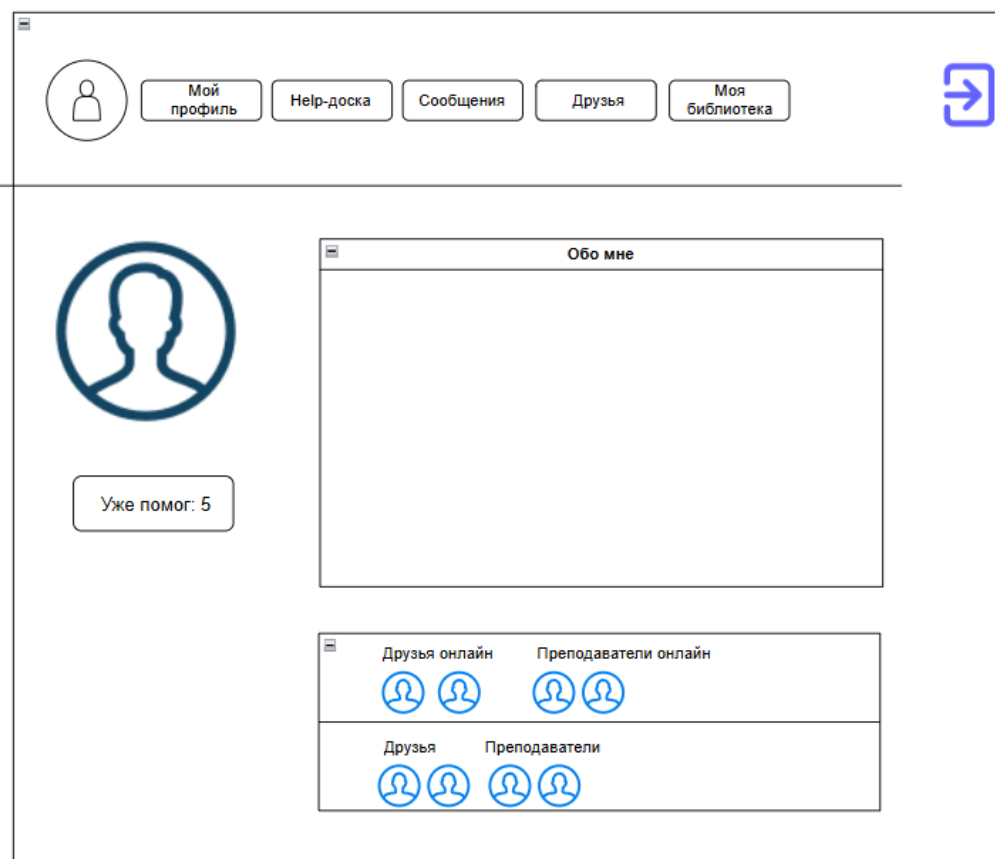


Авторизация

Логин

Пароль

Рисунок Б.1 – Окно авторизации на сервисе



Мой профиль Help-доска Сообщения Друзья Моя библиотека

Обо мне

Уже помог: 5

Друзья онлайн	Преподаватели онлайн
 	 
Друзья	Преподаватели
 	 

Рисунок Б.2 – Окно вкладки «Мой профиль»

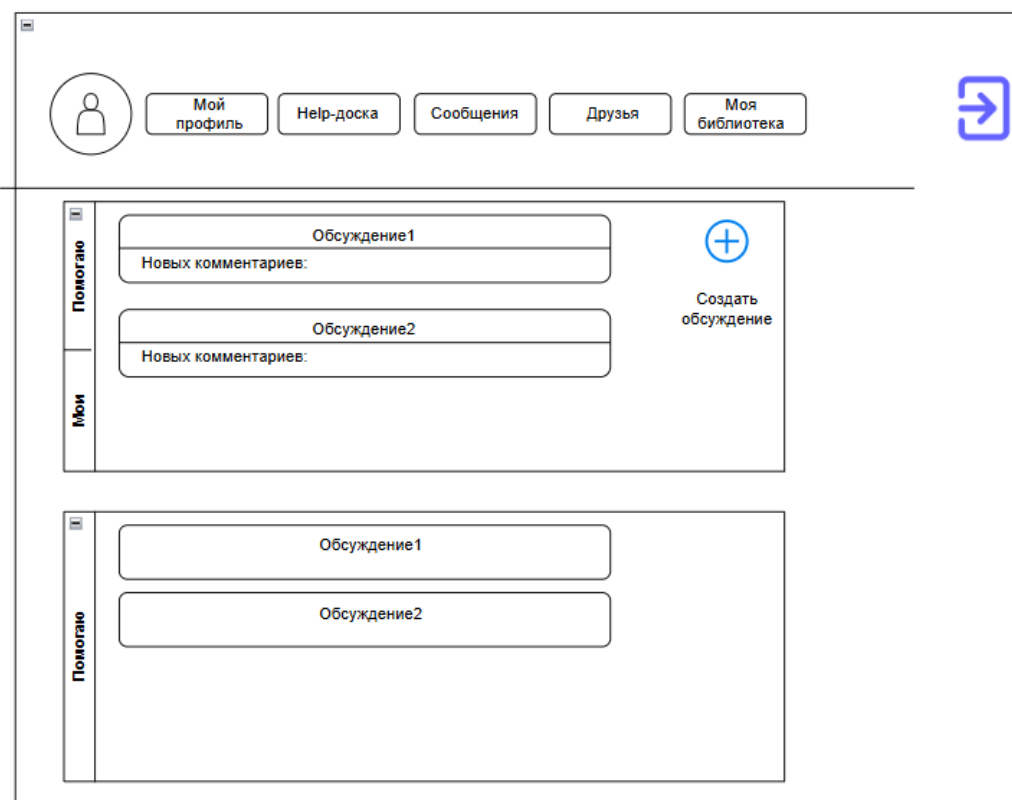


Рисунок Б.3 – Окно вкладки «Help-доска»

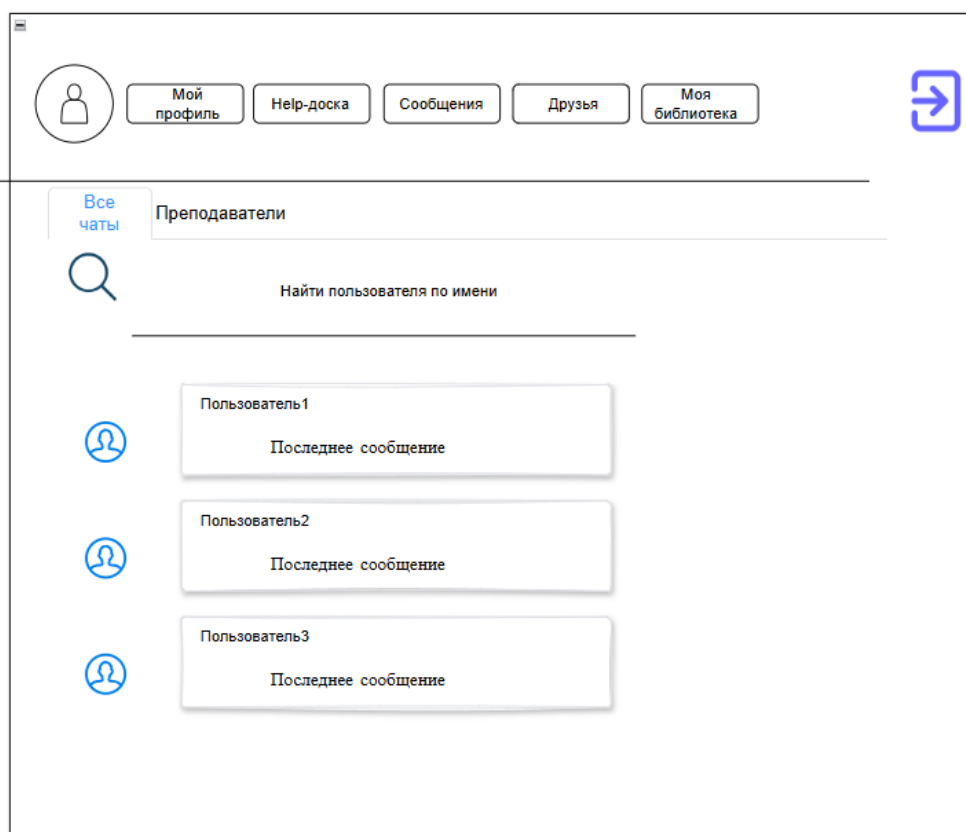


Рисунок Б.4 – Окно вкладки «Сообщения»

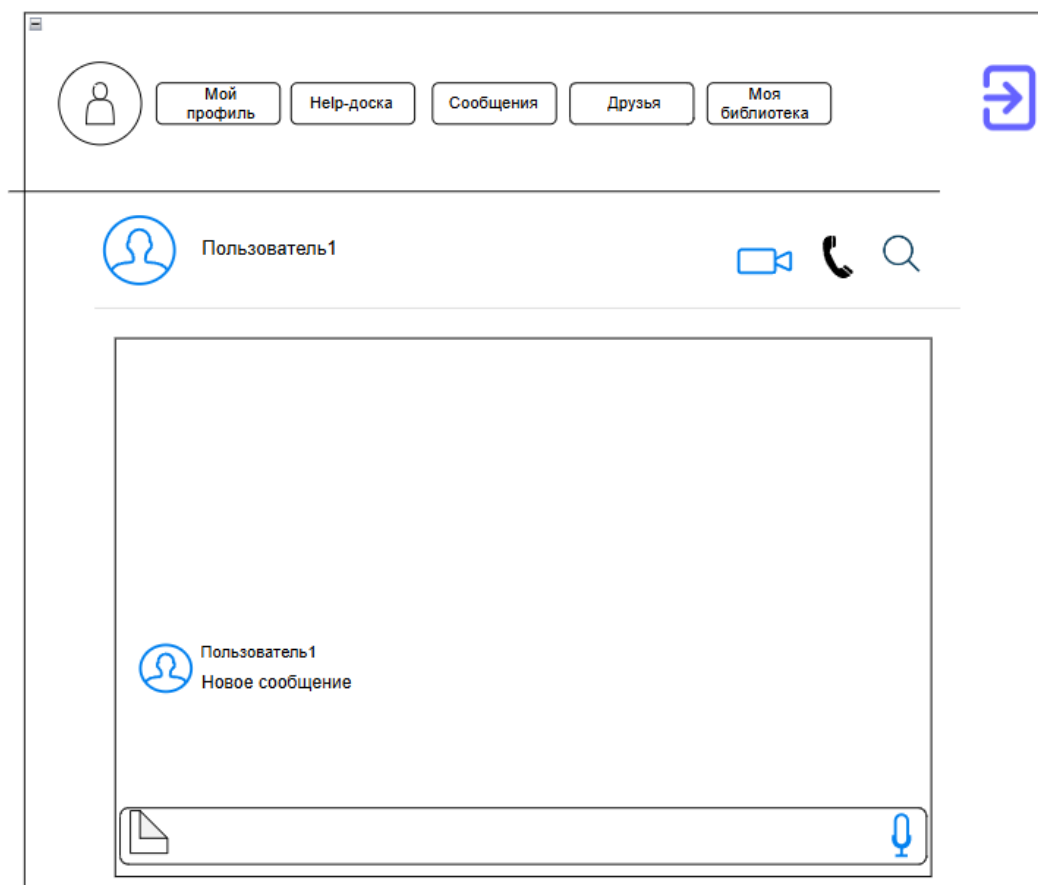


Рисунок Б.5 – Окно открытого чата с пользователем

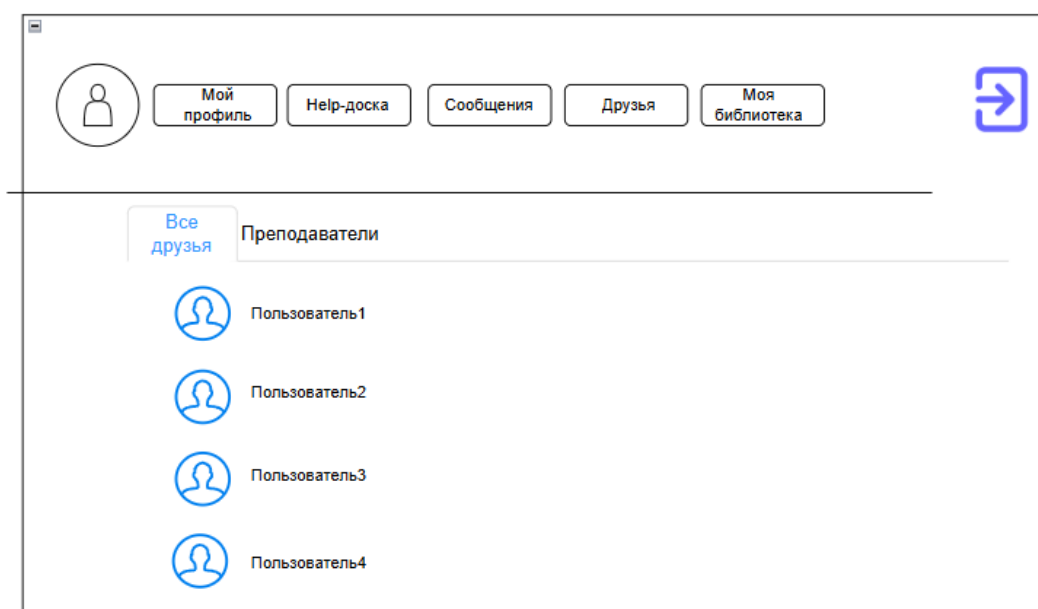


Рисунок Б.6 – Окно вкладки «Друзья»

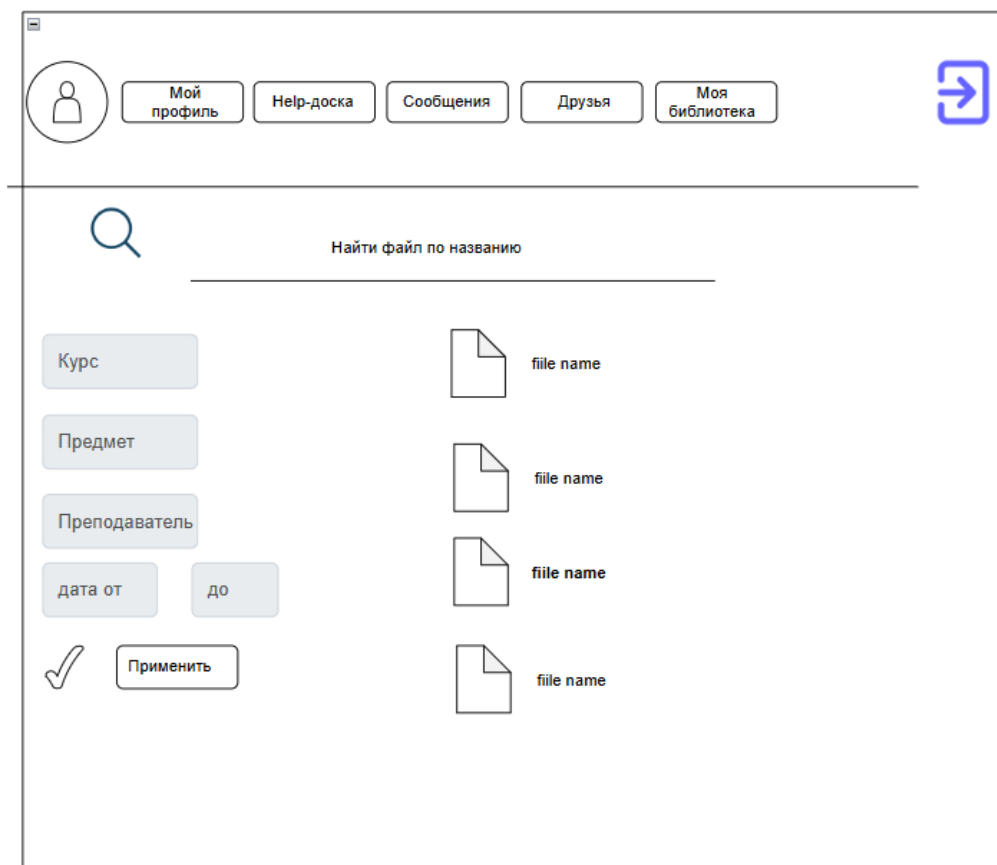


Рисунок Б.7 – Окно вкладки «Моя библиотека»

Приложение В
(рекомендуемое)
Код разработки базы данных

Листинг В.1 – Код создания таблиц базы данных

```

Hibernate: create table chat_users_table (chat_id bigint not null,
      user_id bigint not null)

Hibernate: create table chats_table (count_users_in_chat integer,
      private_chat boolean, chat_creator_id bigint,
      id bigint generated by default as identity,
      subject_id bigint, primary key (id))

Hibernate: create table chats_table_messages (chat_id bigint not null,
      messages_id bigint not null unique)

Hibernate: create table discussions_table (unread_message_count integer,
      discussion_chat_id bigint unique,
      id bigint generated by default as identity, owner_id bigint,
      subject_id bigint, description varchar(255) not null,
      term varchar(255) not null, title varchar(255) not null,
      primary key (id))

Hibernate: create table discussions_table_user_list
      (discussion_id bigint not null, user_list_id bigint not null)

Hibernate: create table files_table (created timestamp(6),
      file_owner_id bigint,
      id bigint generated by default as identity,
      file_name varchar(255), file_original_name varchar(255),
      file_type varchar(255), subject varchar(255),
      term varchar(255), file_content blob, primary key (id))

Hibernate: create table login_data_table
      (id bigint generated by default as identity,
      password varchar(30) not null,
      email varchar(255) unique, primary key (id))

Hibernate: create table messages_table
      (file_in_message_id bigint unique,
      id bigint generated by default as identity,
      message_author_id bigint, message_text varchar(255),
      primary key (id))

Hibernate: create table news_table (author_id bigint,
      id bigint generated by default as identity,
      content varchar(255),
      title varchar(255), primary key (id))

Hibernate: create table subjects_table
      (id bigint generated by default as identity,
      subject_name varchar(255), primary key (id))

Hibernate: create table users_table (friend_list_count integer,
      client_login_data_id bigint unique,
      id bigint generated by default as identity,
      role varchar(255), user_name varchar(255) unique,
      avatar blob, primary key (id))

Hibernate: create table users_table_friend_list
      (friend_list_id bigint not null,

```

```
user_id bigint not null)
```

Листинг В.2 – Код создания связей между таблицами

```
Hibernate: alter table if exists chat_users_table add constraint
FKt7n9fry45x9ybudhe6wbxjib6 foreign key (user_id) references
users_table
Hibernate: alter table if exists chat_users_table add constraint
FKqo12kbxnh35wxy4k242ha6ukv foreign key (chat_id) references
chats_table
Hibernate: alter table if exists chats_table add constraint
FKiayurf5mbjsyplmqvt89371ca foreign key (chat_creator_id)
references users_table
Hibernate: alter table if exists chats_table add constraint
FK3tly70s6uscxxkg7x24tmov98 foreign key (subject_id)
references subjects_table
Hibernate: alter table if exists chats_table_messages add constraint
FKqy9e8oal7hx7i2elali4opig0 foreign key (messages_id)
references messages_table
Hibernate: alter table if exists chats_table_messages add constraint
FKjc73j989xtmd32hanvgqcxo7e foreign key (chat_id) references
chats_table
Hibernate: alter table if exists discussions_table add constraint
FKcyim7ya2w3a51pu3rr9h2no0v foreign key (discussion_chat_id)
references chats_table
Hibernate: alter table if exists discussions_table add constraint
FKmhusirfrmgst0se2v0bui9frk foreign key (owner_id) references
users_table
Hibernate: alter table if exists discussions_table add constraint
FKay7ffap41vr di6bbmhv2bn7gs foreign key (subject_id)
references subjects_table
Hibernate: alter table if exists discussions_table_user_list add
constraint FK4jxdgrrybvd yq9saypy0m2bvb foreign key
(user_list_id) references users_table
Hibernate: alter table if exists discussions_table_user_list add
constraint FK9dep3idsd2vq136k6amipvh2b foreign key
(discussion_id) references discussions_table
Hibernate: alter table if exists files_table add constraint
FK5a00x3pcsmgngf4wuvq7lrcty foreign key (file_owner_id)
references users_table
Hibernate: alter table if exists messages_table add constraint
FKg4bl3yd12fdni9ndfxygouhci foreign key (message_author_id)
references users_table
Hibernate: alter table if exists messages_table add constraint
FKrbg5kuf61g8fgul5c760wbquq foreign key (file_in_message_id)
references files_table
Hibernate: alter table if exists news_table add constraint
FKsd0udb01l642vcbfwqmm6fofu foreign key (author_id) references
users_table
Hibernate: alter table if exists users_table add constraint
FKjf3ba8npny3kvbeutnnupxnxk foreign key (client_login_data_id)
references login_data_table
Hibernate: alter table if exists users_table_friend_list add constraint
FK1jygl4r71w485s2kjdo3psyla foreign key (friend_list_id)
references users_table
Hibernate: alter table if exists users_table_friend_list add constraint
FKn0rv8cylgrheahxl9mgftrjt foreign key (user_id) references
```

users_table

Листинг В.3 – Заполнение таблицы chat_users_table

```
insert into chat_users_table (chat_id, user_id) values (1,1)
insert into chat_users_table (chat_id, user_id) values (2,2)
insert into chat_users_table (chat_id, user_id) values (3,7)
insert into chat_users_table (chat_id, user_id) values (4,5)
insert into chat_users_table (chat_id, user_id) values (5,4)
insert into chat_users_table (chat_id, user_id) values (6,4)
insert into chat_users_table (chat_id, user_id) values (7,4)
insert into chat_users_table (chat_id, user_id) values (8,6)
insert into chat_users_table (chat_id, user_id) values (9,3)
insert into chat_users_table (chat_id, user_id) values (10,1)
```

Листинг В.4 – Заполнение таблицы chats_table

```
insert into chats_table (private_chat, chat_creator_id,
                        count_users_in_chat, subject_id)
values (TRUE,1,2,1)
values (FALSE,1,2,3)
values (TRUE,5,2,2)
values (TRUE,2,3,1)
values (FALSE,4,10,4)
values (FALSE,3,4,5)
values (TRUE,1,5,1)
values (TRUE,2,7,1)
```

Листинг В.5 – Заполнение таблицы chats_table_messages

```
insert into chats_table_messages (chat_id, messages_id) values (1,3)
insert into chats_table_messages (chat_id, messages_id) values (1,4)
insert into chats_table_messages (chat_id, messages_id) values (1,5)
insert into chats_table_messages (chat_id, messages_id) values (1,6)
insert into chats_table_messages (chat_id, messages_id) values (1,7)
insert into chats_table_messages (chat_id, messages_id) values (1,8)
insert into chats_table_messages (chat_id, messages_id) values (1,9)
insert into chats_table_messages (chat_id, messages_id) values (1,10)
```

Листинг В.6 – Заполнение таблицы discussions_table

```
insert into discussions_table (discussion_chat_id, unread_message_count,
                              owner_id, subject_id, description, term,
                              title)
values (1, 6, 5, 4, "Здравствуйте! Помогите решить проблему с 1С. При
запуске срабатывает ошибка","1С","Не работает 1С")
values (2, 3, 2, 1, "Подскажите, какую наиболее правильную связь сделать
в ДВИ (скрин)","ИСИС","Связь в ДВИ")
values (4, 0, 1, 3, "В маткаде неправильно рассчитывается формула. Уже не
знаю, что делать","Mathcad","Неверное значение формулы в
маткаде")
values (3, 14, 3, 4, "Делаю программу на Java, на консоль выводится
ошибка 1245","Java","Ошибка 1245 в Java")
values (6, 7, 2, 2, "Где в ArgoUML находятся компоненты диаграммы
развертывания?","ArgoUML","Д-маРазвертывания в ArgoUML")
```

Листинг В.7 – Заполнение таблицы discussions_table_user_list

```
insert into discussions_table_user_list (discussion_id, user_list_id)
```

```

        values (1, 4)
insert into discussions_table_user_list (discussion_id, user_list_id)
        values (3, 3)
insert into discussions_table_user_list (discussion_id, user_list_id)
        values (1,5)
insert into discussions_table_user_list (discussion_id, user_list_id)
        values (2,4)
insert into discussions_table_user_list (discussion_id, user_list_id)
        values (1,7)
insert into discussions_table_user_list (discussion_id, user_list_id)
        values (2,6)
insert into discussions_table_user_list (discussion_id, user_list_id)
        values (4,1)
insert into discussions_table_user_list (discussion_id, user_list_id)
        values (2,8)

```

Листинг В.8 – Заполнение таблицы login_data_table

```

insert into login_data_table (password, email) values
("asdtmg335", "valentina.egorova.2003@gmail.com")
insert into login_data_table (password, email) values
("g8g89f89", "kattrin1987@mail.ru")
insert into login_data_table (password, email) values
("ajkb8ehjf89", "tyvyvy@yandex.ru")
insert into login_data_table (password, email) values
("Kg63747585", "sws@mail.ru")
insert into login_data_table (password, email) values
("nata6a7a8", "nataly2000@gmail.com")

```

Листинг В.9 – Заполнение таблицы files_table

```

insert into files_table (created, file_owner_id, file_name,
                        file_original_name, file_type, term,
                        file_content blob)
values ("13.05.2024", 1, "ИСИС_Лаб1", "ИСИС_Лаб1.docx", "docx",
        "1 лаба по ИСИС", D:\ИСИС_Лаб1.docx)
values ("15.05.2024", 1, "ИСИС_Лаб2", "ИСИС_Лаб2.docx", "docx",
        "2 лаба по ИСИС", D:\ИСИС_Лаб2.docx)
values ("20.05.2024", 2, "БЗиЭС_1", "БЗиЭС_1.docx", "docx",
        "1 лаба по БЗиЭС", D:\БЗиЭС_1.docx)
values ("19.05.2024", 3, "ДВИ", "ДВИ.png", "png", "ДВИ по ВКР",
        D:\ДВИ.png)
values ("10.05.2024", 4, "ДДеятельности", "ДДеятельности.png", "png",
        "ДДеятельности по ВКР", D:\ДДеятельности.png)

```

Листинг В.10 – Заполнение таблицы messages_table

```

insert into messages_table (file_in_message_id, message_author_id,
                           message_text)
values (1, 1, "Привет")
values (3, 2, "Как дела?")
values (4, 1, "Что делаешь?")
values (5, 4, "Приветствую")
values (6, 7, "Хай, все хорошо")
values (7, 3, "Здравствуйте, Дмитрий Анатольевич!")
values (8, 4, "Привет")
values (9, 6, "Сегодня уезжаю")
values (10, 5, "Вот, нашла лабы по Java")

```

```
values (11, 4, "))))")
```

Листинг В.11 – Заполнение таблицы news_table

```
insert into news_table (author_id, content, title)
values (1, "Сегодня в Пензе +15. Обещают грозу",
        "Погода в Пензе 20.05.2024")
values (2, "Завтра занятия переносятся в аудиторию 7б-202",
        "Смена аудитории")
values (3, "В топ языков программирования в 2024 году вошли...", "Топ-5")
```

Листинг В.12 – Заполнение таблицы subjects_table

```
insert into subjects_table (subject_name varchar(255)) values
("Программирование")
insert into subjects_table (subject_name varchar(255)) values ("1С")
insert into subjects_table (subject_name varchar(255)) values ("Java")
insert into subjects_table (subject_name varchar(255)) values
("Диаграммы UML")
insert into subjects_table (subject_name varchar(255)) values
("Качество ИС")
```

Листинг В.13 – Заполнение таблицы users_table

```
insert into users_table (friend_list_count, client_login_data_id, role,
                        user_name, avatar)
values (58, 2, "Студент", "Валентина Егорова", D:\ava1.png)
values (105, 3, "Студент", "Калашникова Анна", D:\ava2.png)
values (39, 4, "Преподаватель", "Шибанов Сергей ")
values (88, 5, "Преподаватель", "Денис Такташкин", D:\ava4.png)
values (58, 6, "Преподаватель", "Афонин Александр", D:\ava5.png)
```

Листинг В.14 – Заполнение таблицы users_table_friend_list

```
insert into users_table_friend_list (friend_list_id, user_id) values
(1,1)
insert into users_table_friend_list (friend_list_id, user_id) values
(2,2)
insert into users_table_friend_list (friend_list_id, user_id) values
(3,5)
insert into users_table_friend_list (friend_list_id, user_id) values
(4,6)
insert into users_table_friend_list (friend_list_id, user_id) values
(5,8)
```

Приложение Г
(рекомендуемое)
Листинг серверной части сервиса

Листинг Г.1 – Код файла Chat.java модуля Models

```
package com.example.social_network.models;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import java.util.ArrayList;

@Getter
@Setter
@NoArgsConstructor
@JsonIgnoreProperties({"hibernateLazyInitializer"})
@Entity
@Table(name="chats_table")
public class Chat {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    Long id;
    @Column
    Boolean privateChat = true;

    @Column(name = "count_users_in_chat")
    int countUsers = 0;

    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name="chat_creator_id")
    User creator;

    @ManyToMany(cascade = CascadeType.PERSIST, fetch =
        FetchType.EAGER)
    @JoinTable(name = "chat_users_table",
        joinColumns = @JoinColumn(name="chat_id"),
        inverseJoinColumns = @JoinColumn(name="user_id"))
    List<User> users = new ArrayList<>();

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER,
        orphanRemoval = true)
    List<Message> messages = new ArrayList<>();

    @ManyToOne(fetch = FetchType.EAGER)
    Subject subject;

    public void addUser(User user) {
        this.users.add(user);
        this.countUsers += 1;
    }

    public void removeUser(User user) {
        this.users.remove(user);
        this.countUsers -= 1;
    }
}
```

```

    public void addMessage(Message message) {
        this.messages.add(message);
    }

    public void removeMessage(Message message) {
        this.messages.remove(message);
    }

    public Chat(User creator) {
        this.creator = creator;
        addUser(creator);
    }

    public Chat(Boolean privateChat, User creator, Subject
        subject) {
        this.privateChat = privateChat;
        this.creator = creator;
        this.subject = subject;
    }

    //for private chat only
    public User getOtherUser() {
        if (privateChat) {
            for (User user : this.users) {
                if (!user.getId().equals(this.creator.getId())) {
                    return user;
                }
            }
        }
        return null;
    }
}

```

Листинг Г.2 – Код файла Discussion.java модуля Models

```

package com.example.social_network.models;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@JsonIgnoreProperties({"hibernateLazyInitializer"})
@Table(name = "discussions_table")
public class Discussion {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column(nullable = false)
    String title;

    @Column(nullable = false)
    String description;

    @Column(nullable = false)
    String term;
}

```

```

@Column
Integer unreadMessageCount;

@ManyToOne(fetch = FetchType.EAGER)
Subject subject;

@OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
@JoinColumn(name="discussion_chat_id")
Chat chat;

@ManyToOne(fetch = FetchType.EAGER)
User owner;

@ManyToMany(fetch = FetchType.EAGER)
List<User> userList;

public void increaseUnreadMessageCount() {
    this.unreadMessageCount++;
}

public void dropUnreadMessageCount() {
    this.unreadMessageCount = 0;
}

public void addUserToDiscussion(User user) {
    this.userList.add(user);
}

public Discussion(String title, String description, String
    term, Subject subject, Chat chat, User owner) {
    this.title = title;
    this.description = description;
    this.term = term;
    this.subject = subject;
    this.chat = chat;
    this.owner = owner;
    this.unreadMessageCount = 0;
}
}

```

Листинг Г.3 – Код файла File.java модуля Models

```

package com.example.social_network.models;

import jakarta.persistence.*;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.core.io.ByteArrayResource;

@Data
@NoArgsConstructor
@Entity
@Table(name = "files_table")
public class File {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

```

```

@Column
String fileName;

@Column
String fileType;

@Column
String fileOriginalName;

@Column
String term;

@Column
String subject;

@Column
Date created;

@ManyToOne(cascade = CascadeType.PERSIST)
@JoinColumn(name = "file_owner_id")
User fileOwner;

@Lob
byte[] fileContent;

public File(MultipartFile file, User fileOwner) {
    try {
        this.fileContent = file.getBytes();
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    this.fileOwner = fileOwner;
    this.fileOriginalName = file.getOriginalFilename();
    this.fileName =
        file.getOriginalFilename().split("\\.")[0];
    this.fileType =
        file.getOriginalFilename().split("\\.")[1];
    this.created = new Date();
}

public Resource convertToResource() {
    return new ByteArrayResource(this.fileContent) {
        @Override
        public String getFilename() {
            return fileOriginalName;
        }
    };
}

public String getCreatedString() {
    return new SimpleDateFormat("dd-MM-
        yyyy").format(this.created);
}

@Override
public String toString() {

```

```

        return "File{" +
            "id=" + id +
            ", fileName='" + fileName + '\'' +
            ", fileType='" + fileType + '\'' +
            ", fileOriginalName='" + fileOriginalName + '\'' +
            ", term='" + term + '\'' +
            ", subject='" + subject + '\'' +
            ", created=" + created +
            ", fileOwner=" + fileOwner +
            '}' +
        '}';
    }
}

```

Листинг Г.4 – Код файла LoginData.java модуля Models

```

package com.example.social_network.models;

@Getter
@Setter
@NoArgsConstructor
@Entity
@Table(name="login_data_table")
public class LoginData {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    Long id;

    @Email
    @Column(unique = true, name="email")
    String email;

    @NotEmpty
    @NotNull
    @Length(min=3, max=30)
    @Column(nullable = false, name = "password")
    String password;

    public LoginData(String email, String password) {
        this.email = email;
        this.password = password;
    }
}

```

Листинг Г.5 – Код файла Message.java модуля Models

```

package com.example.social_network.models;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import jakarta.persistence.*;

@Getter
@Setter
@NoArgsConstructor
@JsonIgnoreProperties({"hibernateLazyInitializer"})
@Entity
@Table(name="messages_table")

```

```

public class Message {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    Long id;

    @ManyToOne (cascade = CascadeType.PERSIST, fetch =
        FetchType.EAGER)
    @JoinColumn(name="message_author_id")
    User author;

    @Column(name="message_text")
    String text;

    @OneToOne
    @JoinColumn(name="file_in_message_id")
    File file;

    public Message(User author, String text) {
        this.author = author;
        this.text = text;
    }

    public Message(User author, String text, File files) {
        this.author = author;
        this.text = text;
        this.file = files;
    }
}

```

Листинг Г.6 – Код файла News.java модуля Models

```

package com.example.social_network.models;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;

@Entity
@Table(name="news_table")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class News {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column
    String title;

    @Column
    String content;

    @ManyToOne(fetch = FetchType.EAGER)
    User author;
}

```

```

        public News(String title, String content, User author) {
            this.title = title;
            this.content = content;
            this.author = author;
        }
    }
}

```

Листинг Г.7 – Код файла Subject.java модуля Models

```

package com.example.social_network.models;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@NoArgsConstructor
@AllArgsConstructor
@JsonIgnoreProperties({"hibernateLazyInitializer"})

@Entity
@Table(name = "subjects_table")
public class Subject {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column
    String subjectName;

    //TODO teachers
    public Subject(String subjectName) {
        this.subjectName = subjectName;
    }
}

```

Листинг Г.8 – Код файла User.java модуля Models

```

package com.example.social_network.models;

@Getter
@Setter
@JsonIgnoreProperties({"hibernateLazyInitializer"})
@NoArgsConstructor
@Entity
@Table(name = "users_table")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    Long id;

    @Column(unique = true, name = "user_name")
    String userName;

    @Column

```

```

String role;

@Column(name = "friend_list_count")
int friendListCount = 0;

@OneToOne(cascade = CascadeType.ALL, orphanRemoval = true)
@JoinColumn(name = "client_login_data_id")
LoginData loginData;

@JsonIgnore
@ManyToMany(mappedBy = "users", fetch = FetchType.EAGER)
List<Chat> chats = new ArrayList<>();

@JsonIgnore
@ManyToMany(fetch = FetchType.EAGER, cascade =
    CascadeType.PERSIST)
List<User> friendList = new ArrayList<>();

@Lob
byte[] avatar;

public void addUserToFriendList(User user) {
    this.friendList.add(user);
    this.friendListCount += 1;
}

public void removeUserFromFriendList(User user) {
    this.friendList.remove(user);
    this.friendListCount -= 1;
}

public void addUserToChat(Chat chat) {
    this.chats.add(chat);
}

public void removeUserFromChat(Chat chat) {
    this.chats.remove(chat);
}

public String generateBase64Img() {
    return Base64.encodeBase64String(this.avatar);
}

public User(String userName, LoginData loginData, String role) {
    this.userName = userName;
    this.loginData = loginData;
    this.role = role;
}
}

```

Листинг Г.9 – Код файла ChatService.java модуля Services

```

package com.example.social_network.services;

import com.example.social_network.models.Chat;
import com.example.social_network.models.File;
import com.example.social_network.models.Message;
import com.example.social_network.models.User;

```

```

import com.example.social_network.repository.ChatRepository;
import com.example.social_network.repository.FileRepository;
import com.example.social_network.repository.MessageRepository;
import com.example.social_network.repository.UserRepository;

@Service
public class ChatService {

    @Autowired
    private ChatRepository chatRepository;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private MessageRepository messageRepository;

    @Autowired
    private FileRepository fileRepository;

    public void sendMessage(Long userId, Long chatId, String
        messageText, MultipartFile addFile) {
        User user = userRepository.getReferenceById(userId);
        File file = fileRepository.save(new File(addFile, user));
        Message message = messageRepository.save(new
            Message(user, messageText, file));
        Chat chat = chatRepository.getReferenceById(chatId);
        chat.addMessage(message);
        chatRepository.save(chat);
    }

    public void sendMessage(Message message, Long chatId) {
        message = messageRepository.save(message);
        Chat chat = chatRepository.getReferenceById(chatId);
        chat.addMessage(message);
        chatRepository.save(chat);
    }

    public void addUserToChat(Long userId, Long chatId) {
        User user = userRepository.getReferenceById(userId);
        Chat chat = chatRepository.getReferenceById(chatId);
        chat.addUser(user);
        chatRepository.save(chat);
    }
}

```

Листинг Г.10 – Код файла DiscussionService.java модуля Services

```

package com.example.social_network.services;
package com.example.social_network.services;

import com.example.social_network.models.*;
import com.example.social_network.repository.ChatRepository;
import com.example.social_network.repository.DiscussionRepository;
import com.example.social_network.repository.SubjectRepository;
import com.example.social_network.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;

```

```

@Service
public class DiscussionService {

    @Autowired
    private DiscussionRepository discussionRepository;

    @Autowired
    private UserService userService;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private ChatRepository chatRepository;

    @Autowired
    private SubjectRepository subjectRepository;

    @Autowired
    private ChatService chatService;

    public Discussion createNewDiscussion(Long creatorId, String
        title, String term, String description,
        String subjectName) {
        Subject subject =
            subjectRepository.findBySubjectName(subjectName).
            get();
        User owner = userRepository.getReferenceById(creatorId);
        Chat chat = chatRepository.save(new Chat(false, owner,
            subject));
        return discussionRepository.save(new Discussion(title,
            description, term, subject, chat, owner));
    }

    public void addUserToDiscussion(Long discussionId, Long
        userId) {
        User user = userRepository.getReferenceById(userId);
        Discussion discussion =
            discussionRepository.getReferenceById(discussionId);
        chatService.addUserToChat(user.getId(),
            discussion.getChat().getId());
        discussion.addUserToDiscussion(user);
        discussionRepository.save(discussion);
    }

    public void sendMessageToDiscussion(Long discussionId, Long
        userId, String messageText) {
        Discussion discussion =
            discussionRepository.getReferenceById(discussionId);
        chatService.sendMessage(userId,
            discussion.getChat().getId(), messageText);
        if (!discussion.getOwner().getId().equals(userId)) {
            discussion.increaseUnreadMessageCount();
        }
        discussionRepository.save(discussion);
    }
}

```

```

public void sendMessageToDiscussion(Long discussionId, Long
    userId, String messageText, MultipartFile attachment) {
    Discussion discussion =
        discussionRepository.getReferenceById(discussionId);
    chatService.sendMessage(userId,
        discussion.getChat().getId(), messageText, attachment);
    if (!discussion.getOwner().getId().equals(userId)) {
        discussion.increaseUnreadMessageCount();
    }
    discussionRepository.save(discussion);
}

public Discussion openDiscussion(Long discussionId, Long
    userId) {
    Discussion discussion =
        discussionRepository.getReferenceById(discussionId);
    if (discussion.getOwner().getId().equals(userId)) {
        discussion.dropUnreadMessageCount();
    }
    return discussionRepository.save(discussion);
}
}

```

Листинг Г.11 – Код файла NewsService.java модуля Services

```

package com.example.social_network.services;

@Service
public class NewsService {

    @Autowired
    NewsRepository newsRepository;

    @Autowired
    UserRepository userRepository;

    @Autowired
    UserService userService;

    public News createNews(String title, String content, Long
        clientId) {
        return newsRepository.save(new News(title, content,
            userRepository.getReferenceById(clientId)));
    }
}

```

Листинг Г.12 – Код файла UserService.java модуля Services

```

package com.example.social_network.services;

import com.example.social_network.models.*;
import com.example.social_network.repository.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import javax.management.RuntimeErrorException;
import java.io.IOException;
import java.util.List;

```

```

@Service
public class UserService {

    @Autowired
    UserRepository userRepository;

    @Autowired
    LoginDataRepository loginDataRepository;

    @Autowired
    MessageRepository messageRepository;

    @Autowired
    ChatRepository chatRepository;

    @Autowired
    private FileRepository fileRepository;

    public User createUser(String email, String password, String
        name, String role) {
        LoginData loginData = loginDataRepository.save(new
            LoginData(email, password));
        System.out.println(loginDataRepository.getReferenceById
            (1L));
        return userRepository.save(new User(name, loginData,
            role));
    }

    public boolean askLogin(String email, String password) {
        return loginDataRepository.findByEmailAndPassword(email,
            password) != null;
    }

    public boolean askLogin(LoginData loginData) {
        return
            userRepository.existsByLoginDataEmailAndLoginDataPas
            sword(loginData.getEmail(),
                loginData.getPassword());
    }

    public User getByLoginData(LoginData loginData) {
        return
            userRepository.findByLoginDataEmailAndLoginDataPassw
            ord(loginData.getEmail(), loginData.getPassword());
    }

    public User addAvatar(Long userId, MultipartFile avatar) {
        User user = userRepository.getReferenceById(userId);
        try {
            user.setAvatar(avatar.getBytes());
        } catch (IOException e) {
            System.out.println("Ошибка добавления файла: " +
                avatar);
        }
        return userRepository.save(user);
    }
}

```

```

public Chat createChat(Long userId) {
    User author = userRepository.getReferenceById(userId);
    return chatRepository.save(new Chat(author));
}

public Chat startPersonalChat(Long userId, Long friendId) {
    Chat chat = createChat(userId);
    User friend = userRepository.getReferenceById(friendId);
    chat.addUser(friend);
    return chatRepository.save(chat);
}

public Chat addUserToChat(Long userId, Long chatId) {
    User user = userRepository.getReferenceById(userId);
    Chat chat = chatRepository.getReferenceById(chatId);
    if (!chat.getUsers().contains(user)) {
        chat.addUser(user);
        return chatRepository.save(chat);
    }
    System.out.println("Пользователь с id " + userId + " уже
        добавлен в чат с id " + chatId);
    return null;
}

public Message sendMessage(Long userId, Long chatId, String
    messageText) {
    User user = userRepository.getReferenceById(userId);
    Message message = messageRepository.save(new
        Message(user, messageText));
    Chat chat = chatRepository.getReferenceById(chatId);
    chat.addMessage(message);
    chatRepository.save(chat);
    return message;
}

public void sendMessage(Message message, Long chatId) {
    message = messageRepository.save(message);
    Chat chat = chatRepository.getReferenceById(chatId);
    chat.addMessage(message);
    chatRepository.save(chat);
}

public Message sendMessage(Long userId, Long chatId, String
    messageText, MultipartFile addFile) {
    User user = userRepository.getReferenceById(userId);
    File file = fileRepository.save(new File(addFile, user));
    Message message = messageRepository.save(new
        Message(user, messageText, file));
    Chat chat = chatRepository.getReferenceById(chatId);
    chat.addMessage(message);
    chatRepository.save(chat);
    return message;
}
}

```

Листинг Г.13 – Код файла AddUserToDiscussionController.java модуля Controllers

```
package com.example.social_network.controllers;

@Controller
public class AddUserToDiscussionController {

    @Autowired
    UserRepository userRepository;

    @Autowired
    DiscussionService discussionService;

    @Autowired
    DiscussionRepository discussionRepository;

    private Long discussionId = 0L;

    @GetMapping("/addUserToDiscussion/{id}")
    public String addUserToDiscussion(@PathVariable Long id,
        HttpSession session, Model model) {
        User user = (User) session.getAttribute("currentUser");
        model.addAttribute("currentUser", user);

        discussionId = id;
        List<User> users =
            userRepository.findAllByIdIsNot(user.getId());
        model.addAttribute("users", users);
        return "add_user_discussion";
    }

    @GetMapping("/addUserToDiscussion/add/{userId}")
    public String addUserToDiscussionAdd(@PathVariable Long
        userId, HttpSession session, Model model) {
        discussionService.addUserToDiscussion(discussionId,
            userId);
        return "redirect:/openDiscussion/" + discussionId;
    }
}
```

Листинг Г.14 – Код файла AdminController.java модуля Controllers

```
package com.example.social_network.controllers;

import com.example.social_network.models.User;
import com.example.social_network.repository.UserRepository;
import com.example.social_network.services.UserService;
import jakarta.servlet.http.HttpSession;
import org.springframework.web.bind.annotation.PathVariable;
import java.util.List;

@Controller
public class AdminController {

    @Autowired
    UserRepository userRepository;
```

```

@Autowired
UserService userService;

@GetMapping("/adminPage")
public String adminPage(HttpSession session, Model model) {
    User user = (User) session.getAttribute("currentUser");
    model.addAttribute("currentUser",
        userRepository.getReferenceById(user.getId()));
    List<User> users = userRepository.findAll();
    model.addAttribute("usersList", users);
    return "admin_page";
}

@GetMapping("/deleteUser/{userId}")
public String deleteUser(@PathVariable Long userId) {
    User user = userRepository.getReferenceById(userId);
    userRepository.delete(user);
    return "redirect:/adminPage";
}
}

```

Листинг Г.15 – Код файла ChatPageController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.File;
import com.example.social_network.repository.ChatRepository;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.multipart.MultipartFile;
import java.io.IOException;

@Controller
public class ChatPageController {

    @Autowired
    UserService userService;

    @Autowired
    UserRepository userRepository;

    @Autowired
    ChatRepository chatRepository;

    private Long currentChatId = 0L;

    @Autowired
    private FileRepository fileRepository;

    @GetMapping("/chat/{chatId}")
    public String chat(@PathVariable Long chatId, Model model,
        HttpSession session) {
        User user = (User) session.getAttribute("currentUser");
        model.addAttribute("currentUser",
            userRepository.getReferenceById(user.getId()));
        currentChatId =

```

```

        chatRepository.getReferenceById(chatId).getId();
        model.addAttribute("chat",
            chatRepository.getReferenceById(chatId));
        return "chat_page";
    }

    @PostMapping("/sendMessage")
    public String sendMess(@RequestParam(name = "addFile")
        MultipartFile addFile, @RequestParam(name = "message") String
        message, HttpSession session) {
        User user = (User) session.getAttribute("currentUser");
        if (addFile != null && !addFile.isEmpty()) {
            userService.sendMessage(user.getId(), currentChatId,
                message, addFile);
            return "redirect:/chat/" + currentChatId;
        }
        userService.sendMessage(user.getId(), currentChatId,
            message);
        return "redirect:/chat/" + currentChatId;
    }
}

```

Листинг Г.16 – Код файла ChatsPageController.java модуля Controllers

```

package com.example.social_network.controllers;

@Controller
public class ChatsPageController {

    @Autowired
    UserService userService;

    @Autowired
    UserRepository userRepository;

    @Autowired
    ChatRepository chatRepository;

    @GetMapping("/chats")
    public String chats(Model model, HttpSession session) {
        User user = (User) session.getAttribute("currentUser");
        model.addAttribute("currentUser",
            userRepository.getReferenceById(user.getId()));
        return "chats_page";
    }
}

```

Листинг Г.17 – Код файла CreateAccountController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.LoginData;
import com.example.social_network.services.UserService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.Errors;

```

```

import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class CreateAccountController {

    @Autowired
    UserService userService;

    @ModelAttribute(name = "loginData")
    public LoginData getClient() {
        return new LoginData();
    }

    @GetMapping("/createAccount")
    public String open() {
        return "create_account_page";
    }

    @PostMapping("/createAccount/new")
    public String createAccount(@Valid LoginData loginData,
        @RequestParam(name = "clientName") String clientName,
        Errors errors, Model model) {
        if (errors.hasErrors()) {
            model.addAttribute("message", "Неверный формат
                данных клиента!");
            return "create_account_page";
        }
        if (userService.createUser(loginData.getEmail(),
            loginData.getPassword(), clientName, "user").getId() !=
            0) {
            model.addAttribute("message", "Пользователь
                добавлен!");
            return "create_account_page";
        }
        model.addAttribute("message", "Ошибка добавления
            пользователя!");
        return "create_account_page";
    }
}

```

Листинг Г.18 – Код файла CreateDiscussionController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.User;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class CreateDiscussionController {

    @Autowired
    UserService userService;

    @Autowired

```

```

    UserRepository userRepository;

    @Autowired
    DiscussionService discussionService;

    @PostMapping("/createDiscussion")
    public String createDiscussion(HttpSession session,
        @RequestParam String title, @RequestParam String
        description, @RequestParam String term, @RequestParam
        String subject) {
        User user = (User) session.getAttribute("currentUser");
        discussionService.createNewDiscussion(user.getId(),
            title, term, description, subject);
        return "redirect:/helpDesk";
    }
}

```

Листинг Г.19 – Код файла CreateNewsController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.News;
import com.example.social_network.models.User;
import com.example.social_network.repository.NewsRepository;
import com.example.social_network.repository.UserRepository;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class CreateNewsController {

    @Autowired
    UserRepository userRepository;

    @Autowired
    private NewsRepository newsRepository;

    @GetMapping("/createNews")
    public String createNews(HttpSession session, Model model) {
        User user = (User) session.getAttribute("currentUser");
        boolean isAdminOrModer =
            user.getRole().equals("moderator") ||
            user.getRole().equals("admin");
        model.addAttribute("isAdminOrModer", isAdminOrModer);
        model.addAttribute("currentUser", user);
        return "create_news_form";
    }

    @PostMapping("/createNews")
    public String createNews(@RequestParam(name = "title") String
        title, @RequestParam(name = "content") String content,
        HttpSession session) {
        User user = (User) session.getAttribute("currentUser");
    }
}

```

```

        News news = new News(title, content, user);
        newsRepository.save(news);
        return "redirect:/newsPage";
    }
}

```

Листинг Г.20 – Код файла DiscussionPageController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.Chat;
import com.example.social_network.repository.UserRepository;
import com.example.social_network.services.DiscussionService;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.multipart.MultipartFile;

@Controller
public class DiscussionPageController {

    @Autowired
    UserRepository userRepository;

    @Autowired
    DiscussionService discussionService;

    @GetMapping("/openDiscussion/{id}")
    public String openDiscussionPage(@PathVariable Long id,
        HttpSession session, Model model) {
        User user = (User) session.getAttribute("currentUser");
        model.addAttribute("currentUser",
            userRepository.getReferenceById(user.getId()));

        Discussion discussion =
            discussionService.openDiscussion(id, user.getId());
        model.addAttribute("discussion", discussion);

        Chat chat = discussion.getChat();
        model.addAttribute("chat", chat);

        return "discussion_page";
    }

    @PostMapping("/sendMessage/discussion/{id}")
    public String sendMessageToDiscussion(@PathVariable Long id,
        @RequestParam(name = "addFile") MultipartFile addFile,
        @RequestParam(name = "message") String message, HttpSession
        session) {

        User user = (User) session.getAttribute("currentUser");
        if (addFile != null && !addFile.isEmpty()) {
            discussionService.sendMessageToDiscussion(id,
                user.getId(), message, addFile);
            return "redirect:/openDiscussion/" + id;
        }
        discussionService.sendMessageToDiscussion(id,
            user.getId(), message);
        return "redirect:/openDiscussion/" + id;
    }
}

```

```
}
```

Листинг Г.21 – Код файла EditNewsController.java модуля Controllers

```
package com.example.social_network.controllers;

import com.example.social_network.models.News;
import com.example.social_network.models.User;
import com.example.social_network.repository.NewsRepository;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class EditNewsController {

    @Autowired
    NewsRepository newsRepository;

    @GetMapping("/editNews/{newsId}")
    public String editNewsForm(@PathVariable(name = "newsId") Long
        newsId, Model model, HttpSession session) {
        model.addAttribute("news",
            newsRepository.getReferenceById(newsId));
        User user = (User) session.getAttribute("currentUser");
        boolean isAdminOrModer =
            user.getRole().equals("moderator") ||
            user.getRole().equals("admin");
        model.addAttribute("isAdminOrModer", isAdminOrModer);
        model.addAttribute("currentUser", user);
        return "edit_news_form";
    }

    @PostMapping("/editNews/{newsId}")
    public String editNewsPost(@PathVariable(name = "newsId") Long
        newsId, @RequestParam(name = "title") String title,
        @RequestParam(name = "content") String content) {
        News news = newsRepository.getReferenceById(newsId);
        news.setTitle(title);
        news.setContent(content);
        newsRepository.save(news);
        return "redirect:/newsPage";
    }
}
```

Листинг Г.22 – Код файла EditUserController.java модуля Controllers

```
package com.example.social_network.controllers;

import com.example.social_network.models.LoginData;
import com.example.social_network.models.User;
import com.example.social_network.repository.LoginDataRepository;
import jakarta.servlet.http.HttpSession;
```

```

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.multipart.MultipartFile;
import java.io.IOException;

@Controller
public class EditUserController {

    @Autowired
    UserService userService;

    @Autowired
    UserRepository userRepository;

    @Autowired
    LoginDataRepository loginDataRepository;

    @GetMapping("/editUser/{userId}")
    public String openEditUser(@PathVariable Long userId, Model
        model, HttpSession session) {
        User user = userRepository.getReferenceById(userId);
        System.out.println("\n\n" + user + "\n\n");
        model.addAttribute("editUser",
            userRepository.getReferenceById(userId));
        user = (User) session.getAttribute("currentUser");
        model.addAttribute("currentUser", user);
        return "edit_user_form";
    }

    @PostMapping("/updateUser/{userId}")
    public String updateUser(@PathVariable Long userId,
        @RequestParam(name = "name") String name,
        @RequestParam(name = "email") String email,
        @RequestParam(name = "password") String password,
        @RequestParam(name = "role") String role) {
        System.out.println("\n\nName: " + name + "\nEmail: " +
            email + "\nPassword: " + password);

        User userUpdate =
            userRepository.getReferenceById(userId);
        if (name != null && !name.isEmpty()) {
            userUpdate.setUserName(name);
            userUpdate = userRepository.save(userUpdate);
        }
        if (email != null && !email.isEmpty()) {
            LoginData loginData =
                loginDataRepository.getReferenceById(userUpdate.get
                    LoginData().getId());
            loginData.setEmail(email);
            loginDataRepository.save(loginData);
        }
        if (password != null && !password.isEmpty()) {
            LoginData loginData =
                loginDataRepository.getReferenceById(userUpdate.get
                    LoginData().getId());
            loginData.setPassword(password);
            loginDataRepository.save(loginData);
        }
    }
}

```

```

        if (role != null && !role.isEmpty()) {
            userUpdate.setRole(role);
            userUpdate = userRepository.save(userUpdate);
        }
        return "redirect:/userPage";
    }

    @PostMapping("/updateAvatar/{userId}")
    public String updateAvatar(@PathVariable Long userId,
                              @RequestParam(name = "avatar") MultipartFile avatar,
                              Model model, HttpSession session) {
        User userUpdate =
            userRepository.getReferenceById(userId);
        if (avatar != null && !avatar.isEmpty()) {
            try {
                userUpdate.setAvatar(avatar.getBytes());
                userRepository.save(userUpdate);
                return "redirect:/editUser/" + userId;
            } catch (IOException e) {
                model.addAttribute("message", "Ошибка загрузки аватара: " + e);
            }
        }
        return openEditUser(userId, model, session);
    }
}

```

Листинг Г.23 – Код файла FilesController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.File;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.Resource;
import org.springframework.http.HttpHeaders;

@Controller
public class FilesController {

    @Autowired
    FileRepository fileRepository;

    @GetMapping("/downloadFile/{id}")
    public ResponseEntity<Resource> downloadFileById(@PathVariable Long id) {
        File file = fileRepository.getReferenceById(id);
        return ResponseEntity.ok().
            header(HttpHeaders.CONTENT_DISPOSITION,
                "attachment; filename=\"" +
                file.getFileOriginalName() + "\"")
            .body(file.convertToResource());
    }
}

```

Листинг Г.24 – Код файла FriendListController.java модуля Controllers

```

package com.example.social_network.controllers;

```

```

import com.example.social_network.models.Chat;
import com.example.social_network.models.User;
import com.example.social_network.repository.UserRepository;
import com.example.social_network.services.UserService;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import java.util.List;
import java.util.Objects;

@Controller
public class FriendListController {

    @Autowired
    UserRepository userRepository;

    @Autowired
    UserService userService;

    @PostMapping("/searchUser")
    public String findUser(@RequestParam(name="searchQuery")
        String searchQuery, Model model, HttpSession session)
    {
        User currentUser = userRepository.getReferenceById(((User)
            session.getAttribute("currentUser")).getId());
        List<User> friends = currentUser.getFriendList();
        List<User> findUsers =
            userRepository.findAllByUserNameContainingIgnoreCase(sea
                rchQuery);
        for (User userFind :
            userRepository.findAllByUserNameContainingIgnoreCase
                (searchQuery)) {
            if (Objects.equals(userFind.getId(),
                currentUser.getId())) {
                findUsers.remove(userFind);
            }
            for (User friend : friends) {
                if (Objects.equals(userFind.getId(),
                    friend.getId())) {
                    findUsers.remove(userFind);
                }
            }
        }
        model.addAttribute("findUsers", findUsers);
        model.addAttribute("searchQuery", searchQuery);
        return openFriendList(session, model);
    }

    @GetMapping("/addUserToFriendList/{userId}")
    private String addUserToFriendList(@PathVariable Long userId,
        HttpSession session) {
        User user = (User) session.getAttribute("currentUser");

```

```

        userService.addUserToFriendList(user.getId(), userId);
        return "redirect:/friends";
    }

    @GetMapping("/chatWithFriend/{friendId}")
    public String chatWithFriend(@PathVariable Long friendId,
        HttpSession session) {
        User user = (User) session.getAttribute("currentUser");
        user = userRepository.getReferenceById(user.getId());
        List<Chat> chats = user.getChats();
        for (Chat chat : chats) {
            for (User chatter : chat.getUsers()) {
                if (Objects.equals(chatter.getId(), friendId)) {
                    return "redirect:/chat/" + chat.getId();
                }
            }
        }
        Chat chat = userService.startPersonalChat(user.getId(),
            friendId);
        return "redirect:/chat/" + chat.getId();
    }
}

```

Листинг Г.25 – Код файла FriendPageController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.User;
import com.example.social_network.repository.UserRepository;

@Controller
public class FriendPageController {

    @Autowired
    UserService userService;

    @Autowired
    UserRepository userRepository;

    @GetMapping("/friendPage/{friendId}")
    public String openFriendPage(@PathVariable Long friendId,
        Model model, HttpSession session) {
        model.addAttribute("currentUser",
            userRepository.getReferenceById(friendId));
        User currentUser = (User)
            session.getAttribute("currentUser");
        model.addAttribute("currentUser", currentUser);
        return "friend_page";
    }
}

```

Листинг Г.26 – Код файла HelpDeskController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.Discussion;
import com.example.social_network.repository.DiscussionRepository;
import com.example.social_network.services.DiscussionService;

```

```

import org.springframework.ui.Model;

@Controller
public class HelpDeskController {

    @Autowired
    private DiscussionService discussionService;

    @Autowired
    private DiscussionRepository discussionRepository;

    @Autowired
    private UserService userService;

    @Autowired
    private UserRepository userRepository;

    @GetMapping("/helpDesk")
    public String helpDesk(Model model, HttpSession session) {
        User user = (User) session.getAttribute("currentUser");
        model.addAttribute("currentUser",
            userRepository.getReferenceById(user.getId()));

        List<Discussion> myDiscussions =
            discussionRepository.findAllByOwner(user);
        List<Discussion> otherDiscussions =
            discussionRepository.findAllByUserListContaining
                (user);
        model.addAttribute("myDiscussions", myDiscussions);
        model.addAttribute("otherDiscussions", otherDiscussions);

        for (Discussion d : myDiscussions) {
            System.out.println("\n\n " + d.getTitle() + "\n\n");
        }
        return "help_desk_page";
    }
}

```

Листинг Г.27 – Код файла LoginController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.LoginData;
import com.example.social_network.services.UserService;
import jakarta.servlet.http.HttpSession;
import jakarta.validation.Valid;
import org.springframework.ui.Model;
import org.springframework.validation.Errors;
import org.springframework.web.bind.annotation.ModelAttribute;

@Controller
public class LoginController {
    @Autowired
    UserService userService;

    @Autowired
    UserRepository userRepository;
}

```

```

@ModelAttribute(name = "loginData")
public LoginData addLoginData() {
    return new LoginData();
}

@GetMapping("")
public String redirectToLogin() {
    return "redirect:/login";
}

@GetMapping("/login")
public String openLogin() {
    return "login_page";
}

@GetMapping("/askLogin")
public String askLogin(@Valid LoginData loginData, Model
    model, Errors errors, HttpSession session) {
    if (errors.hasErrors()) {
        model.addAttribute("message", "Неверный формат
            данных пользователя!");
        return "login_page";
    }
    if (userService.askLogin(loginData)) {
        User user = userService.getByLoginData(loginData);
        session.setAttribute("currentUser", user);
        return "redirect:/userPage";
    }
    model.addAttribute("message", "Ошибка в данных
        пользователя!");
    return "login_page";
}

@GetMapping("/logOut")
public String logOut(HttpSession session) {
    session.invalidate();
    return "redirect:/login";
}
}

```

Листинг Г.28 – Код файла NewsController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.News;
import com.example.social_network.models.User;
import com.example.social_network.repository.NewsRepository;
import com.example.social_network.repository.UserRepository;
import jakarta.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;

@Controller
public class NewsController {

    @Autowired
    UserService userService;

```

```

@Autowired
UserRepository userRepository;

@Autowired
NewsService newsService;

@Autowired
NewsRepository newsRepository;

@GetMapping("/newsPage")
public String newsPage(HttpSession session, Model model) {
    User user = (User) session.getAttribute("currentUser");
    model.addAttribute("currentUser",
        userRepository.getReferenceById(user.getId()));
    boolean isAdminOrModer =
        user.getRole().equals("moderator") ||
        user.getRole().equals("admin");
    model.addAttribute("isAdminOrModer", isAdminOrModer);
    List<News> newsList = newsRepository.findAll();
    Collections.reverse(newsList);
    model.addAttribute("newsList", newsList);
    return "news_page";
}

@GetMapping("/deleteNews/{newsId}")
public String deleteNews(@PathVariable long newsId) {
    newsRepository.deleteById(newsId);
    return "redirect:/newsPage";
}
}

```

Листинг Г.29 – Код файла UserLibraryController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.File;
import com.example.social_network.repository.UserRepository;
import jakarta.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestParam;
import java.text.ParseException;
import java.text.SimpleDateFormat;

@Controller
public class UserLibraryController {

    private final UserRepository userRepository;
    private final FileRepository fileRepository;

    public UserLibraryController(UserRepository userRepository,
        FileRepository fileRepository) {
        this.userRepository = userRepository;
        this.fileRepository = fileRepository;
    }

    @GetMapping("/library")

```

```

public String openLibrary(HttpSession session, Model model) {
    User user = (User) session.getAttribute("currentUser");
    user = userRepository.getReferenceById(user.getId());
    model.addAttribute("currentUser", user);

    List<File> files =
        fileRepository.findAllByFileOwner(user);
    model.addAttribute("files", files);
    return "file_lib";
}

@PostMapping("/files/findByCriteria")
public String findByCriteria(@RequestParam String term,
    @RequestParam String subj, @RequestParam String dateFrom,
    @RequestParam String dateTo, Model model, HttpSession session)
    throws ParseException {
    boolean termExist = term != null && !term.isBlank();
    boolean subjExist = subj != null && !subj.isBlank();
    boolean dateExist = dateFrom != null &&
        !dateFrom.isBlank() && dateTo != null &&
        !dateTo.isBlank();

    Date from = new Date();
    Date to = new Date();

    if (dateExist) {
        SimpleDateFormat formater = new
            SimpleDateFormat("yyyy-MM-dd");
        from = formater.parse(dateFrom);
        to = formater.parse(dateTo);
    }

    List<File> files = new ArrayList<>();

    if (termExist && subjExist && dateExist) {
        files =
            fileRepository.findAllByTermAndSubjectAndCreatedBetween(
                term, subj, from, to);
    } else if (termExist && subjExist) {
        files = fileRepository.findAllByTermAndSubject(term,
            subj);
    } else if (termExist && dateExist) {
        files =
            fileRepository.findAllByTermAndCreatedBetween(
                term, from, to);
    } else if (subjExist && dateExist) {
        files =
            fileRepository.findAllBySubjectAndCreatedBetween(
                subj, from, to);
    } else if (termExist) {
        files = fileRepository.findAllByTerm(term);
    } else if (subjExist) {
        files = fileRepository.findAllBySubject(subj);
    } else if (dateExist) {
        files = fileRepository.findAllByCreatedBetween(from,
            to);
    }
    model.addAttribute("files", files);
}

```

```

        User user = (User) session.getAttribute("currentUser");
        user = userRepository.getReferenceById(user.getId());
        model.addAttribute("currentUser", user);
        return "file_lib";
    }

    @PostMapping("/files/findByName")
    public String findFilesByName(@RequestParam("fileNameSearch")
        String fileNameSearch, HttpSession session, Model model) {
        User user = (User) session.getAttribute("currentUser");
        user = userRepository.getReferenceById(user.getId());
        model.addAttribute("currentUser", user);
        List<File> files =
            fileRepository.findAllByFileNameContainingIgnoreCase
                (fileNameSearch);
        model.addAttribute("files", files);
        if (files.isEmpty()) {
            model.addAttribute("message", "File with name " +
                fileNameSearch + " not found");
        }
        return "file_lib";
    }
}

```

Листинг Г.30 – Код файла UserPageController.java модуля Controllers

```

package com.example.social_network.controllers;

import com.example.social_network.models.User;
import com.example.social_network.repository.UserRepository;

@Controller
public class UserPageController {

    @Autowired
    UserRepository userRepository;

    @GetMapping("/userPage")
    public String openClientPage(HttpSession session, Model model)
    {
        User user = (User) session.getAttribute("currentUser");
        model.addAttribute("currentUser",
            userRepository.getReferenceById(user.getId()));
        return "user_page";
    }
}

```

Листинг Г.31 – Код файла ChatRepository.java модуля Repository

```

package com.example.social_network.repository;

import com.example.social_network.models.Chat;
import com.example.social_network.models.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository

```

```
public interface ChatRepository extends JpaRepository<Chat, Long> {
    List<Chat> findAllByUsersContains(User user);
}
```

Листинг Г.32 – Код файла DiscussionRepository.java модуля Repository

```
package com.example.social_network.repository;

import com.example.social_network.models.Discussion;
import com.example.social_network.models.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface DiscussionRepository extends JpaRepository<Discussion, Long> {

    List<Discussion> findAllByOwner(User author);
    List<Discussion> findAllByUserListContaining(User user);
}
```

Листинг Г.33 – Код файла FileRepository.java модуля Repository

```
package com.example.social_network.repository;

import com.example.social_network.models.File;
import com.example.social_network.models.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Date;
import java.util.List;

@Repository
public interface FileRepository extends JpaRepository<File, Long> {
    List<File> findAllByFileOwner(User fileOwner);
    List<File> findAllByCreatedBetween(Date start, Date end);
    List<File> findAllByFileNameContainingIgnoreCase(String
        fileName);
    List<File> findAllByTerm(String term);
    List<File> findAllBySubject(String subject);
    List<File> findAllByTermAndSubject(String term, String subj);
    List<File> findAllByTermAndCreatedBetween(String term, Date
        start, Date end);
    List<File> findAllBySubjectAndCreatedBetween(String subj, Date
        start, Date end);
    List<File> findAllByTermAndSubjectAndCreatedBetween(String
        term, String subj, Date start, Date end);
}
```

Листинг Г.34 – Код файла LoginDataRepository.java модуля Repository

```
package com.example.social_network.repository;

import com.example.social_network.models.LoginData;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

```
@Repository
public interface LoginDataRepository extends JpaRepository<LoginData,
Long> {
    LoginData findByEmailAndPassword(String email, String
        password);
}
```

Листинг Г.35 – Код файла MessageRepository.java модуля Repository

```
package com.example.social_network.repository;

import com.example.social_network.models.Message;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface MessageRepository extends JpaRepository<Message,
    Long> {
}
```

Листинг Г.36 – Код файла NewsRepository.java модуля Repository

```
package com.example.social_network.repository;

import com.example.social_network.models.News;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface NewsRepository extends JpaRepository<News, Long>
{
}
```

Листинг Г.37 – Код файла SubjectRepository.java модуля Repository

```
package com.example.social_network.repository;

import com.example.social_network.models.Subject;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;

@Repository
public interface SubjectRepository extends JpaRepository<Subject,
    Long> {
    Optional<Subject> findBySubjectName(String name);
}
```

Листинг Г.38 – Код файла UserRepository.java модуля Repository

```
package com.example.social_network.repository;

import com.example.social_network.models.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface UserRepository extends JpaRepository<User, Long> {
```

```
User findByLoginDataEmailAndLoginDataPassword(String email,  
        String password);  
boolean existsByLoginDataEmailAndLoginDataPassword(String  
        email, String password);  
List<User> findAllByUsernameContainingIgnoreCase(String name);  
List<User> findAllByIdIsNot(Long id);}
```

Приложение Д
(рекомендуемое)
Листинг клиентской части сервиса

Листинг Д.1 – Код файла add_user_discussion.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Discussion</title>

    <link rel="stylesheet"
          th:href="@{/webjars/bootstrap/css/bootstrap.min.css}"
          type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
            type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
            type="text/javascript"></script>

</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="align-content-center align-self-center container
border">
    <div class="align-content-center align-self-center container
border-bottom" style="margin-bottom: 20px;"
        th:each="user : ${users}">
        <h5 th:text="${user.getUserName()}"></h5>
        <a th:href="@{/addUserToDiscussion/add/{userId}
(userId=${user.getId()})}">
            Добавить в дискуссию
        </a>
    </div>
</div>
</body>
</html>
```

Листинг Д.2 – Код файла admin_page.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Страница админа</title>

    <link rel="stylesheet"
          th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
            type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
            type="text/javascript"></script>

</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="container align-content-center align-self-center">
    <h6>Пользователи</h6>
    <table class="table table-striped"><thead class="thead-light">
    <tr>
        <th></th> <th>#</th><th>email</th><th>Имя
```

```

пользователя</th><th>Роль</th></tr>
</thead>
<tbody>
<tr th:each="user : ${usersList}">
<td></td>
<td th:text="${user.getId()}"></td>
<td th:text="${user.getLoginData().getEmail()}"></td>
<td th:text="${user.getUserName()}"></td>
<td th:text="${user.getRole()}"></td>
<td><a th:href="@{/deleteUser/{userId}
(userId=${user.getId()})}" style="color: red">Удалить</a>
</td>
<td><a th:href="@{/editUser/{userId}
(userId=${user.getId()})}">Изменить</a></td>
</tr>
</tbody>
</table>
</div>
</body>
</html>

```

Листинг Д.3 – Код файла chat_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<title th:text="'Чат с ' +
${chat.getOtherUser().getUserName()}"></title>

<link rel="stylesheet"
th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" type="text/css"/>
<script th:src="@{/webjars/jquery/jquery.min.js}"
type="text/javascript"></script>
<script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
type="text/javascript"></script>
</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="container align-self-center align-content-center">

<div class="container align-content-center align-self-center
border-bottom" style="margin-bottom: 15px"
th:each="message: ${chat.getMessages()}">
<h5 th:text="${message.getAuthor().getUserName()} + ' написал:'"></h5>
<h6 th:text="${message.getText()}"></h6>
<div th:if="${message.getFile() != null}" class="container">
<a th:text="${message.getFile().getFileName()}"
th:href="@{/downloadFile/{id}
(id=${message.getFile().getId()})}"></a>
</div>
</div>
<div class="container align-self-center align-content-center
border-top border-dark">

```

```

        <form method="post" th:action="@{/sendMessage}"
        enctype="multipart/form-data">
            <input type="text" name="message" required=""
            placeholder="Write smth...">
            <input type="file" name="addFile">
            <button type="submit" class="btn-outline-
            success">Send</button>
        </form>
    </div>
</div>
</body>
</html>

```

Листинг Д.4 – Код файла chats_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta content="width=device-width, initial-scale=1, shrink-to-
    fit=no" name="viewport">
    <title>Чаты</title>

    <link rel="stylesheet"
    th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
    type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
    type="text/javascript"></script>

</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="container align-content-center align-self-center
    border-bottom" style="margin-bottom: 30px"
th:if="${!currentUser.getChats().isEmpty()}" th:each="chat :
    ${currentUser.getChats()}">
    <a th:href="@{/chat/{chatId} (chatId=${chat.getId()})}">
        <h5 th:text="'Чат с ' + ${chat.getOtherUser().getUserName() }
        '></h5>
    </a>
    <div th:if="${!chat.getMessages().isEmpty()}">
        <h6
        th:text="${chat.getMessages().getLast().getAuthor().getUserName() }
        + ' последнее сообщение: '"></h6>
        <h6 th:text="${chat.getMessages().getLast().getText()}"></h6>
    </div>
</div>
</body>
</html>

```

Листинг Д.5 – Код файла create_account_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

```

```

<head>
  <title>Создание аккаунта</title>
  <script th:src="@{/webjars/jquery/jquery.min.js}"
type="text/javascript"></script>
  <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
type="text/javascript"></script>
</head>
<body>
<div class="position-absolute top-20 start-50 translate-middle shadow p-3
mb-5 rounded">

  <form action="#" method="post" th:action="@{/createAccount/new}"
th:object="${loginData}">
    <h1 class="h3 mb-3 font-weight-normal">Введите данные</h1>
    <table class="loginTable">

      <tr>
        <td>
          <input class="form-control" id="inputName"
placeholder="Имя пользователя" required="" name = "clientName"
type="text">
        </td>
      </tr>

      <tr>
        <td>
          <input autofocus="" class="form-control"
id="inputEmail" placeholder="Email пользователя" required=""
th:field="*{email}" type="text"/>
        </td>
      </tr>

      <tr>
        <td>
          <input class="form-control" id="inputPassword"
placeholder="Пароль" required="" th:field="*{password}"
type="text">
        </td>
      </tr>

      <tr>
        <td>
          <button class="btn btn-lg btn-primary btn-
block" type="submit">Создать!</button>
        </td>
      </tr>
    </table>
    <button class="btn-outline-danger btn-block btn">
      <a href="/">Отмена</a>
    </button>
  </form>
</div>

<div class="alert alert-secondary alert-dismissible fade show text-center
message mt-3"
  role="alert"
  h:if="${message != null}">

```

```

        [[${message}]]
        <button aria-label="Close" class="close btn-sm" data-
dismiss="alert" type="button">
            <span aria-hidden="true">×</span>
        </button>
    </div>

</body>
</html>

```

Листинг Д.6 – Код файла create_discussion_form.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Добавить обсуждение</title>

    <script th:src="@{/webjars/jquery/jquery.min.js}"
type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
type="text/javascript"></script>

</head>
<body>
<div class="align-content-center align-self-center container">
    <form method="post" th:action="@{/createDiscussion}">
        <input name="title" type="text" required=""
placeholder="Заголовок">
        <input name="description" type="text" required=""
placeholder="Описание">
        <select name="subject">
            <option value="">Выберите предмет</option>
            <option value="math">Матеша</option>
            <option value="programming">UML</option>
            <option value="patterns">Паттерны</option>
            <option value="SQL">SQL</option>
            <option value="algorithms">Алгоритмы</option>
        </select>
        <button class="btn-primary">Создать</button>
    </form>
</div>
</body>
</html>

```

Листинг Д.7 – Код файла create_news_form.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta content="width=device-width, initial-scale=1, shrink-to-
fit=no" name="viewport">
    <title>Изменить пользователя</title>
    <link rel="stylesheet"
th:href="@{/webjars/bootstrap/css/bootstrap.min.css}" type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
type="text/javascript"></script>

```

```

</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="container align-content-lg-center align-self-center"
style="width: 600px">
    <form th:action="@{/createNews}" method="post">
        <div class="container" style="margin-bottom: 20px">
            <input required="" size="40" placeholder="Заголовок"
            name="title">
        </div>
        <div class="container">
            <textarea required="" rows="8" cols="40"
placeholder="Текст новости" name="content"
maxlength="400"></textarea>
        </div>
        <div class="container align-self-center align-content-lg-
center" style="margin-top: 20px; width: 300px">
            <button type="submit" class="btn btn-outline-
primary">Оправить</button>
        </div>
    </form>
</div>
</body>
</html>

```

Листинг Д.8 – Код файла discussion_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta content="width=device-width, initial-scale=1, shrink-to-
fit=no" name="viewport">
    <title>Discussion</title>

    <link rel="stylesheet"
        th:href="@{/webjars/bootstrap/css/bootstrap.min.css}"
        type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
        type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
        type="text/javascript"></script>

</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="container align-self-center align-content-center">

    <div class="align-self-center align-content-center container
border-bottom border-dark" style="margin-bottom: 20px">
        <h5 th:text="'Тема: ' + ${discussion.getTitle()}"
style="border-bottom: darkgray"></h5><br>
        <h6 th:text="'Предмет: ' +
            ${discussion.getSubject()}"></h6>
        <h6 th:text="'Семестр: ' + ${discussion.getTerm()}"></h6>
        <h6 th:text="'${discussion.getDescription()}"></h6>
    </div>

```

```

<div class="container align-self-end">
    <a th:href="@{/addUserToDiscussion/{id}
        (id=${discussion.getId()})}">Добавить
        пользователя</a>
</div>

<div class="container align-content-center align-self-center
border-bottom" style="margin-bottom: 15px"
th:each="message: ${chat.getMessages()}">
    <h5 th:text="${message.getAuthor().getUserName()} + '
        написал: '"></h5>
    <h6 th:text="${message.getText()}"></h6>
    <div th:if="${message.getFile() != null}"
        class="container">
        <a th:text="${message.getFile().getFileName()}"
            th:href="@{/downloadFile/{id}
                (id=${message.getFile().getId()})}"></a>
    </div>
</div>

<div class="container align-self-center align-content-center
border-top border-dark" style="margin-top: 20px; margin-
bottom: 20px">
    <form method="post"
        th:action="@{/sendMessage/discussion/{id}
            (id=${discussion.getId()})}"
        enctype="multipart/form-data">
        <input type="text" name="message" required=""
            placeholder="Write smth...">
        <input type="file" name="addFile">
        <button type="submit" class="btn-outline-
            success">Send</button>
    </form>
</div>
</div>
</body>
</html>

```

Листинг Д.9 – Код файла edit_news_form.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta content="width=device-width, initial-scale=1, shrink-to-
fit=no" name="viewport">
    <title>Редиктировать новость</title>
    <link rel="stylesheet"
        th:href="@{/webjars/bootstrap/css/bootstrap.min.css}"
        type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
type="text/javascript"></script>

</head>
<body>
<div th:replace="fragments/header::header"></div>
<div class="container align-content-center align-self-center">

```

```

        <form method="post" th:action="@{/editNews/{id}}
(id=${news.getId()})">
            <table>
                <tbody>
                    <tr><td>
                        <input th:placeholder="${news.getTitle()}"
                            type="text" name="title">
                    </td></tr>
                    <tr><td>
                        <textarea th:placeholder="${news.getContent()}"
                            cols="40" rows="5" name="content"></textarea>
                    </td></tr>
                    <tr><td>
                        <button type="submit" class="btn-
                            primary">Сохранить</button>
                    </td></tr>
                </tbody>
            </table>
        </form>
    </div>

</body>
</html>

```

Листинг Д.10 – Код файла edit_user_form.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
        type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
        type="text/javascript"></script>
</head>
<body>
<div th:replace="fragments/header::header"></div>

    <div class="row border-bottom">
        <div class="col-md-5 border-right" style="width: 300px">
            <div class="align-items-center text-center p-3
                py-5">
                <h4 class="text-left align-text-top">Обновите
                    аватар</h4>
                
                <div class="container align-self-center">
                    <form method="post"
                        th:action="@{/updateAvatar/{userId}}
                        (userId=${editUser.getId()})"
                        enctype="multipart/form-data">
                        <input id="input-file" type="file"
                            name="avatar" class="form-control-file"
                            accept="multipart/form-data"/>
                        <button class="btn btn-sm btn-outline-
                            success float-sm-left" type="submit">
                            Загрузить
                        </button>
                    </form>
                </div>
            </div>
        </div>
    </div>

```

```

        </form>
    </div>
</div>
<div class="col-md-auto">
    <form method="post" th:action="@{/updateUser/{userId}
        (userId=${editUser.getId()})}">
    <h4 class="text-left">Измените данные пользователя</h4>
    <table>
        <tbody>
            <tr>
                <td>
                    <input th:placeholder=
                        "${editUser.getUserName()}"
                        type="text" name="name"
                        maxlength="50">
                </td>
            </tr>
            <tr>
                <td>
                    <input th:placeholder=
                        "${editUser.getLoginData().getEmail()}"
                        type="text" name="email"
                        maxlength="30">
                </td>
            </tr>
            <tr>
                <td>
                    <input th:placeholder=
                        "${editUser.getLoginData().getPassw
                        ord()}" type="text" name="password"
                        maxlength="50">
                </td>
            </tr>
            <tr th:if="${currentUser.getRole().equals
                ('admin')}">
                <td>
                    <input th:placeholder=
                        "${editUser.getRole()}" type="text"
                        name="role"
                        maxlength="50">
                </td>
            </tr>
            <tr>
                <td>
                    <button type="submit" class="btn-
                        outline-
                        success">Подтвердить</button>
                </td>
            </tr>
        </tbody>
    </table>
    </form>
</div>
</div>
<div class="alert alert-secondary alert-dismissible fade show text-center
message mt-3"

```

```

        role="alert"
        th:if="{message != null}">
        [{message}]
        <button aria-label="Close" class="close btn-sm" data-
dismiss="alert" type="button">
            <span aria-hidden="true">×</span>
        </button>
</div>
</body>
</html>

```

Листинг Д.11 – Код файла file_lib.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Библиотека</title>

    <link rel="stylesheet"
        th:href="@{/webjars/bootstrap/css/bootstrap.min.css}"
        type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
        type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
        type="text/javascript"></script>

</head>
<body>

<div class="container align-content-center align-self-center
border-bottom border-dark">
    <form th:action="@{/files/findByName}" method="post"
        style="margin-bottom: 20px">
        <input type="text" name="fileNameSearch"
            placeholder="Название" style="outline-width: 0; width:
200px">
    </form>
</div>

<div class="container align-content-center align-self-center"
    style="margin-top: 20px;margin-bottom: 20px">
    <div class="row">
        <div class="col-md-5 border-right">
            <form th:action="@{/files/findByCriteria}"
                method="post">

                <select name="subj">
                    <option value="">Выберите предмет</option>
                    <option value="math">Java</option>
                    <option value="programming">UML</option>
                    <option value="patterns">Паттерны</option>
                    <option value="SQL">SQL</option>
                    <option value="algs">Алгоритмы</option>
                </select>

                <div class="container align-content-center">
                    <table>
                        <tbody>

```

```

        <tr>
            <td>
                <p>Дата от: </p>
                <input type="date"
                    name="dateFrom"
                    style="margin-bottom:
                        10px">
            </td>
            <td>
                <p>до: </p>
                <input type="date"
                    name="dateTo"
                    style="margin-bottom:
                        10px">
            </td>
        </tr>
    </tbody>
</table>
</div>
<button class="btn-outline-success"
    type="submit">Find</button>
</form>
</div>
<div class="col-md">
<div class="container align-self-center align-content-
    center border-bottom"
    th:each="file : ${files}">
    <div>Add pick</div>
    <a th:href="@{/downloadFile/{id} (id =
        ${file.getId()})}"
        ><p th:text="${file.getFileName()}"></p></a>
    </div>
</div>
</div>
</div>
<div class="alert alert-secondary alert-dismissible fade show
    text-center message mt-3"
    role="alert"
    th:if="${message != null}">
    [[${message}]]
    <button aria-label="Close" class="close btn-sm" data-
        dismiss="alert" type="button">
        <span aria-hidden="true">x</span>
    </button>
</div>
</body>
</html>

```

Листинг Д.12 – Код файла friend_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Профиль пользователя</title>

    <script th:src="@{/webjars/jquery/jquery.min.js}"
        type="text/javascript"></script>

```

```

        <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
type="text/javascript"></script>

</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="container align-content-center align-self-center rounded">
    <div class="row border-bottom">
        <div class="col-md-3 border-right">
            <div class="d-flex flex-column align-items-center
text-center p-3 py-5">
                
            </div>
        </div>
        <div class="col-md">
            <h4 class="text-left">Профиль пользователя</h4>
            <h6 th:text="'Имя пользователя: ' +
${currentUser.getUserName()}'"/>
            <h6 th:text="'Email: ' +
${currentUser.getLoginData().getEmail()}'"/>
        </div>
    </div>
</div>
</body>

```

Листинг Д.13 – Код файла friends_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Профиль пользователя</title>

    <link rel="stylesheet"
th:href="@{/webjars/bootstrap/css/bootstrap.min.css}"
type="text/css"/>
    <script th:src="@{/webjars/jquery/jquery.min.js}"
type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
type="text/javascript"></script>

</head>
<body>

<div class="container align-self-center align-content-center">
    <form th:action="@{/searchUser}" method="post">
        <div class="input-group rounded">
            <input name="searchQuery" type="search" class="form-
control rounded" placeholder="Найти" aria-
label="Search"
aria-describedby="search-addon"/>
            <span class="input-group-text border-0">
                <button type="submit">Найти</button>
            </span>
        </div>
    </form>

```

```

</div>

<div th:if="${findUsers != null}">
    <div th:if="${findUsers.isEmpty()}"
        class="align-content-center align-content-center
            container border-bottom " style="margin-bottom: 20px">
        <h5 th:text="'Пользователь ' + ${searchQuery} + ' не
            найден!'"></h5>
    </div>

    <div th:if="${!findUsers.isEmpty()}"
        class="align-content-center align-content-center
            container border-bottom " style="margin-bottom: 20px">
        <div class="col-sm-6">
            <div class="list list-row block border-bottom"
                style="margin-bottom: 15px"
                th:each="user : ${findUsers}">
                <div class="list-item">
                    <div><a th:href="@{/friendPage/{friendId}}
                        (friendId=${user.getId()})}" data-
                        abc="true"><span
                            class="w-48 avatar gd-primary">
                            </span></a>
                    </div>
                    <div class="flex">
                        <a th:href="@{/friendPage/{friendId}}
                            (friendId=${user.getId()})}"
                            th:text="${user.getUserName()}"
                            class="item-author text-color" data-
                            abc="true"></a>
                    </div>
                    <a th:href="@{/addUserToFriendList/{id}}
                        (id=${user.getId()})}">Добавить в
                        друзья</a>
                </div>
            </div>
        </div>
    </div>
</div>

<div class="col-sm-6">
<div class="list list-row block border-bottom" style="margin-
    bottom: 15px"
    th:each="friend : ${currentUser.getFriendList()}">
    <div class="list-item">
        <div><a th:href="@{/friendPage/{friendId}}
            (friendId=${friend.getId()})}" data-
            abc="true"><span
                class="w-48 avatar gd-primary">
                </span></a>
        </div>
    </div>

```

```

        <div class="flex">
            <a th:href="@{/friendPage/{friendId}
                (friendId=${friend.getId()})}"
                th:text="${friend.getUserName()}"
                class="item-author text-color" data-
                abc="true"></a>
        </div>
        <div class="container">
            <button class="btn-outline-success"><a
                th:href="@{/chatWithFriend/{userId}
                (userId=${friend.getId()})}"
                th:text="'B чат с ' +
                    ${friend.getUserName()}"></a></button>
        </div>
    </div>
</div>

</body>
</html>

```

Листинг Д.14 – Код файла help_desk_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Help desc</title>

    <script th:src="@{/webjars/jquery/jquery.min.js}"
        type="text/javascript"></script>
    <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
        type="text/javascript"></script>
</head>
<body>
<div th:replace="fragments/header::header"></div>

<div class="container align-self-center align-content-center border">
    <a href="/createDiscussion">create new</a>
</div>
<div>
    <h6>Moi helps</h6>
    <div th:if="${!myDiscussions.isEmpty()}">
        <div th:each="myDiscussion : ${myDiscussions}"
            class="container align-content-center align-self-
            center border-bottom" style="margin-bottom:
            20px; margin-top: 10px">
            <a th:href="@{/openDiscussion/{id}
                (id=${myDiscussion.getId()})}">
                <h6 th:text="${myDiscussion.getTitle()}"></h6>
            </a>
            <p
                th:text="${myDiscussion.getUnreadMessageCount()}"><
            /p>
        </div>
    </div>
</div>

```

```

<div>
  <h6>Others</h6>
  <div th:if="${!otherDiscussions.isEmpty()}">
    <div th:if="${!otherDiscussions.isEmpty()}"
      th:each="otherDiscussion : ${otherDiscussions}"
      class="container align-content-center align-self-
        center border-bottom" style="margin-bottom: 20px;
        margin-top: 10px">
      <a th:href="@{/openDiscussion/{id}}
        (id=${otherDiscussion.getId()})">
        <h6
          th:text="${otherDiscussion.getTitle()}">
        </h6>
      </a>
    </div>
  </div>
</div>
</body>
</html>

```

Листинг Д.15 – Код файла login_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Log in</title>

  <script th:src="@{/webjars/jquery/jquery.min.js}"
    type="text/javascript"></script>
  <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
    type="text/javascript"></script>

</head>
<body>
  <form action="#" method="get" th:action="@{/askLogin}"
    th:object="${loginData}">
    <h1 class="h3 mb-3 font-weight-normal">Введите
      данные</h1>
    <table class="loginTable">
      <tr>
        <td>
          <label class="sr-only"
            for="inputPassword">Password</label>
          <input class="form-control" id="inputPassword"
            placeholder="Password" required=""
            th:field="*{password}"
            type="password">
        </td>
      </tr>
      <tr>
        <td>
          <button class="btn btn-lg btn-primary btn-
            block" type="submit">Войти</button>
        </td>
      </tr>
    </table>
  </form>
  <p><a href="/createAccount">Создать аккаунт!</a></p>

```

```

</div>

<div class="alert alert-secondary alert-dismissible fade show
  text-center message mt-3"
  role="alert"
  h:if="{message != null}">
  [{message}]
  <button aria-label="Close" class="close btn-sm" data-
    dismiss="alert" type="button">
    <span aria-hidden="true">×</span>
  </button>
</div>
</body>
</html>

```

Листинг Д.16 – Код файла news_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta content="width=device-width, initial-scale=1, shrink-to-
    fit=no" name="viewport">
  <title>Новости</title>

  <script th:src="@{/webjars/jquery/jquery.min.js}"
    type="text/javascript"></script>
  <script th:src="@{/webjars/bootstrap/js/bootstrap.min.js}"
    type="text/javascript"></script>

</head>
<body>

<div class="container align-self-center align-content-center
  border-bottom" style="margin-top: 20px"
  th:if="{!newsList.isEmpty()}" th:each="news : {newsList}">
  <h5 th:text="{news.getTitle()}" style="border-bottom:
    darkgray"></h5>
  <h6 th:text="{news.getContent()}"></h6>
  <br>
  <h6 th:text="'Автор: ' +
    {news.getAuthor().getUserName()}"></h6>
  <div th:if="{isAdminOrModer}"
    class="container" style="margin-top: 10px; margin-bottom:
      10px">
    <a th:href="@{/deleteNews/{id} (id={news.getId()})}"
      style="color: red">Удалить</a>
  </div>
  <div th:if="{isAdminOrModer}"
    class="container" style="margin-top: 10px; margin-bottom:
      10px">

    <a th:href="@{/editNews/{id} (id={news.getId()})}"
      style="color: green">Редактировать</a>

  </div>
</div>
</body>
</html>

```

Листинг Д.17 – Код файла user_page.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    
<div class="col-md">
    <h4 class="text-left">Профиль пользователя</h4>
    <h6 th:text="'Имя пользователя: ' +
        ${currentUser.getUserName()}" />
    <h6 th:text="'Email: ' +
        ${currentUser.getLoginData().getEmail()}" />
    <button class="btn btn-outline-primary profile-
        button btn-link" type="button">
        <a th:href="@{/editUser/{userId}
            (userId=${currentUser.getId()})}">Изменить
            профиль</a>
    </button>
</div>
</div>
</div>
</body>
</html>

```