

МИНИСТЕРСТВО НАУКИ ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет «Вычислительная техника» (наименование)	Кафедра «Системы автоматизированного проектирования» (наименование)
Направление подготовки	09.03.01 Информатика и вычислительная техника (код и наименование)
Профиль	Системы автоматизированного проектирования (наименование)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА
на тему:

Система онлайн-записи студентов на консультации к преподавателям

Студент	<div></div> <div>(подпись, дата)</div>	<div>Нагорная Д.А.</div> <div>(фамилия, инициалы)</div>	
Руководитель	<div></div> <div>(подпись, дата)</div>	<div>Гудков А.А.</div> <div>(фамилия, инициалы)</div>	
Нормоконтролёр	<div>проф. каф. САПР</div> <div>(должность, место работы)</div>	<div></div> <div>(подпись, дата)</div>	<div>Бождай А.С.</div> <div>(фамилия, инициалы)</div>

Работа допущена к защите

Заведующий кафедрой	 (подпись)	 (фамилия, инициалы)
---------------------	---------------	-------------------------

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

Секретарь ГЭК	 (подпись)	 (фамилия, инициалы)
---------------	---------------	-------------------------

“Утверждаю”

Заведующий кафедрой САПР

/И.П. Бурукина/

“ ” 2025 г.

ЗАДАНИЕ

на выполнение бакалаврской работы

1. Студент гр. 21BBS1 факультета ВТ направления Информатика и вычислительная техника
Нагорная Дарья Александровна
(фамилия, имя, отчество)
2. Руководитель ВКР _____ к.т.н., доцент кафедры САПР Гудков Алексей Анатольевич
3. Время выполнения бакалаврской работы с 08.04.2025 г. по 08.06.2025 г.
4. Место преддипломной практики кафедра САПР ФВТ ПГУ
5. Тема бакалаврской работы: «Система онлайн-записи студентов на консультации к преподавателям»

Тема утверждена приказом ПГУ № 603/0 от “ 06 ” мая 2025 г.

6. Техническое задание на бакалаврскую работу: предмет и назначение разработки, функциональные, программные, технические требования и т.п.
- 6.1. Назначение разработки: Разработать систему онлайн-записи студентов на консультации к преподавателям, предназначенную для автоматизации процесса организации расписания консультаций преподавателями, записи студентов на консультации, а также формирования отчета об успеваемости для руководства (деканата). Цель разработки – упростить и обеспечить удобное взаимодействие между участниками образовательного процесса. Система должна быть реализована в виде веб-приложения.
- 6.2. Требования к функциональности системы: Система будет объединять три типа пользователей: преподаватель, студент и руководство. Приложение должно разрешать преподавателю создавать консультации с указанием настроек расписания, на которые может записываться студент с указанием причины записи, и вести учет консультаций. В обязанности руководства будут входить контроль успеваемости студентов и оценка работы преподавателей, для выявления затруднений в учебном процессе и принятия эффективных решений исходя из аналитики. Приложение должно осуществлять авторизацию пользователя на основе уже имеющейся базы данных, при которой, в зависимости от роли, участники системы смогут получить доступ к своим функциональным возможностям, включающая редактирование личных данных.
- 6.3. Требования к программному обеспечению: Серверная часть веб-приложения должна быть реализована с использованием следующих технологий: языка программирования Java 22 (или другой совместимой версии); веб-сервера Tomcat; СУБД PostgreSQL; графического интерфейса pgAdmin; фреймворка Spring Framework; модулей Spring Boot, Spring Security, Spring Data JPA, Spring MVC. Клиентская часть веб-приложения должна функционировать в любом браузере, поддерживающем JavaScript (ES6+), HTML5, CSS3.
- 6.4. Требования к аппаратному обеспечению: Серверная часть предъявит минимальные требования: процессор 2-ядерный CPU 2ГГц+, ОЗУ 4Гб+, HDD 10 Гб+ свободного места и сетевое подключение. Клиентская часть должна быть совместима с любым устройством, поддерживающим работу веб-браузера.

7. Объем и содержание основной части проекта

7.1. Содержание пояснительной записки бакалаврской работы: перечень вопросов, подлежащих разработке, расчетов, обоснований, описаний

7.1.1 Анализ технического задания и предметной области

7.1.2 Обзор используемых технологий и программных средств

7.1.3 Описание разработанного программного обеспечения

7.1.4 Описание пользовательского интерфейса

8. Календарный график работ по выполнению проекта.

Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя, консультанта
1. Анализ предметной области	5 %	08.04-15.04	
2. Анализ технического задания	15 %	15.04-16.04	
3. Проектирование ПО	30 %	16.04-20.04	
4. Разработка ПО	35%	20.04-10.05	
5.Тестирование ПО	5%	10.05-15.05	
6. Оформление ПЗ	10 %	15.05-30.05	
7. Нормоконтроль		31.05-08.06	

Дата выдачи задания

«__» _____ 20__

Руководитель бакалаврской работы

Гудков Алексей Анатольевич
(фамилия, имя, отчество)

Задание к исполнению принял

«__» _____ 20__

Студент

Нагорная Дарья Александровна
(фамилия, имя, отчество)

Бакалаврскую работу к защите допустить

Декан ФВТ _____ / Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка к выпускной квалификационной работе содержит 77 страниц, 48 рисунков, 24 таблицы, 11 источников и 1 приложение.

СИСТЕМА ОНЛАЙН-ЗАПИСИ СТУДЕНТОВ, КОНСУЛЬТАЦИИ ПРЕПОДАВАТЕЛЕЙ, JAVA, TOMCAT, POSTGRESQL, PGADMIN 4, SPRING FRAMEWORK, SPRING BOOT, SPRING SECURITY, SPRING DATA JPA, SPRING MVC, INTELLIJ IDEA, APACHE MAVEN, JAVASCRIPT, HTML, CSS.

Цель работы – реализовать систему онлайн-записи студентов на консультации к преподавателям. Данная система позволит улучшить качество образовательного процесса: снизить нагрузку на преподавателей за счет создания гибких консультаций с возможностью настройки расписания под свои потребности; повысить уровень активности, вовлеченности и ответственности за собственное обучение у студентов из-за автоматизации записи на консультации; оперативно отслеживать динамику образовательного процесса благодаря формированию отчетов с анализом работы преподавателей и успеваемости студентов, а именно посещаемости и сдаче задолженностей.

Результатом работы является веб-приложение для онлайн-записи студентов на консультации к преподавателям. Данное приложение было разработано в соответствии с требованиями, предъявленными в техническом задании на бакалаврскую работу.

Для разработки программного продукта использовался широкий технологический стек, к которому относятся: языки программирования Java, JavaScript; язык разметки HTML; язык стилей CSS; веб-сервер Tomcat; система управления базами данных PostgreSQL; графический интерфейс pgAdmin 4; фреймворк Spring Framework; модули Spring Boot, Spring Security, Spring Data JPA, Spring MVC; интегрированная среда разработки IntelliJ IDEA; инструмент для сборки проекта Apache Maven.

					ВКР 090301.06 25 81 01			
Изм.		№ докум.	Подпись	Дата				
Разраб.		Нагорная Д.А.			Система онлайн-записи студентов на консультации к преподавателям Пояснительная записка	Лит.	Лист	Листов
Провер.		Гудков А.А.					4	
Реценз.						гр. 21BBC1		
Н. Контр.		Бождай А.С.						
Утверд.		Бурукина И.П.						

Содержание

Перечень обозначений и сокращений.....	7
Введение.....	8
1 Анализ предметной области и технического задания.....	10
1.1 Анализ предметной области	10
1.1.1 Анализ актуальности темы	10
1.1.2 Анализ аналогов	12
1.2 Анализ технического задания.....	15
1.2.1 Постановка задачи	15
1.2.2 Анализ требований к серверной и клиентской части на разработку.....	16
2 Проектирование программного обеспечения	18
2.1 Выбор технологий и инструментов разработки	18
2.1.1 Выбор языка программирования для серверной части.....	18
2.1.2 Выбор языка программирования для клиентской части.....	19
2.1.3 Выбор фреймворка и его модулей	20
2.1.4 Выбор базы данных	21
2.1.5 Выбор графического интерфейса для управления базой данных.....	23
2.1.6 Выбор среды разработки	24
2.2 Архитектура клиент-серверного приложения	25
2.3 Проектирование базы данных	27
2.3.1 Инфологическое проектирование	28
2.3.2 Логическое проектирование	30
2.3.3 Физическое проектирование.....	34
2.4 Моделирование структуры поведения системы	39
3 Разработка программного обеспечения.....	46
3.1 Создание БД.....	46
3.2 Подготовка среды разработки	48
3.3 Реализация структуры проекта.....	48
3.4 Выбор архитектурного паттерна	49

4 Тестирование программного обеспечения	51
4.1 Создание и удаление консультации преподавателем	51
4.2 Создание и удаление записи студентом	57
4.3 Просмотр статистики студентов и преподавателей руководством	62
4.4 Адаптация под мобильные устройства	63
Заключение	65
Список используемых источников.....	66
Приложение	67

Перечень обозначений и сокращений

В настоящей бакалаврской работе применяют следующие сокращения и обозначения:

ЭИОС – электронная информационно-образовательная среда.

ПО – программное обеспечение.

ЯП – язык программирования.

ОС – операционная система.

ЭИОС – электронная информационно-образовательная среда.

IDE – интегрированная среда разработки.

HTML (HyperText Markup Language) – язык гипертекстовой разметки.

CSS (Cascading Style Sheets) – формальный язык стилей для описания внешнего вида документа.

SQL (Structured Query Language) – язык структурированных запросов для работы с базами данных.

XML (eXtensible Markup Language) – расширяемый язык разметки.

HTTP (Hypertext Transfer Protocol) – протокол передачи гипертекстовых данных.

REST (Representational State Transfer) – стиль архитектуры для распределенных систем.

SOAP (Simple Object Access Protocol) – протокол обмена структурированными сообщениями.

MVC (Model-View-Controller) – паттерн проектирования, разделяющий приложение на три компонента.

CSRF (Cross-Site Request Forgery) – атака, при которой злоумышленник заставляет браузер выполнять действия на доверенном сайте.

XSS (Cross-Site Scripting) – атака, заключающаяся во внедрении вредоносного кода на веб-страницу.

БД – база данных.

СУБД – система управления базами данных.

JPA (Java Persistence API) – API для сохранения Java-объектов в реляционные БД.

Введение

В настоящее время веб-приложения являются мощным инструментом для автоматизации бизнес-процессов. Пользователи выполняют определенные задачи и функции с использованием веб-интерфейса, позволяющего клиенту отправлять запросы на сервер, который, в свою очередь, взаимодействует с базой данных для хранения и извлечения необходимой информации. Таким образом, веб-приложение – это приложение, имеющее клиент-серверную архитектуру. Пользователь через клиентскую часть приложения, выступающая в роли браузера, отправляет запросы на сервер, выступающий в качестве веб-сервера, который их обрабатывает, обращаясь к базе данных, и передает клиенту.

У веб-приложений много преимуществ [1], одним из которых является отсутствие необходимости установки, поскольку работают в браузере, и обновлений, которые происходят автоматически на сервере, что упрощает поддержку и эксплуатацию, повышает надежность и обеспечивает более эффективное и безопасное функционирование приложений.

Следующим преимуществом считается то, что веб-приложения предъявляют более гибкие требования к конечному пользователю, так как позволяют им быть независимым от браузера, аппаратной платформы и операционной системы. Таким образом, пользователи могут получить доступ к веб-приложению с разного браузера, устройства и ОС.

Другое преимущество – облегчение организации централизованного хранения данных, при которой все данные хранятся в едином месте на одном или нескольких централизованных серверах, что позволяет организовать более эффективное управление данными и обеспечить их безопасность.

В учебных образовательных организациях важным вопросом всегда выступала успеваемость студентов. В университетах основными проблемами являются наличие академических учебных задолженностей, а также необходимость обсуждения курсовых, дипломных работ, научных статей под руководством преподавателя и многое другое. Поэтому основная потребность заключается в организации консультаций, которые помогут справиться с

					ВКР 090301.06 25 81 01	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

проблемами студентов. Для этого сделаны расписания консультаций и приема сдачи задолженностей, развешенные на кафедрах, что есть во многих университетах. Но довольно часто все участники образовательного процесса сталкиваются с проблемами в проведении консультаций: неконтролируемое количество и неявка студентов, отсутствие прозрачности и доступности консультаций, сложности учета и аналитики для деканата, гибкость расписания и многое другое.

Разработка системы онлайн-записи студентов на консультации к преподавателям [2], которая будет реализована в виде веб-приложения, поможет справиться с приведенными выше трудностями, поскольку позволит оптимизировать нагрузку преподавателей, сделать удобным запись студентов и устранить очереди на консультации, а также отслеживать посещаемость и контролировать сдачи задолженностей студентов, что важно знать деканату, и решать прочие проблемы, описанные в следующем разделе.

					ВКР 090301.06 25 81 01	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области и технического задания

Прежде чем приступить к разработке программного продукта, необходимо провести анализ предметной области и технического задания для четкого изложения целей и задач, направленных на устранение существующих трудностей. Это первый этап при разработке программного обеспечения, включающего в себя описание проблемы, сбор всей доступной информации о системе, рассмотрение аналогов и постановку требований.

1.1 Анализ предметной области

Анализ предметной области связан с определением потребностей и характеристик разрабатываемого ПО. Здесь важно ясное представление о том, как организована предметная область без разрабатываемого продукта и кто будет потребителем и какие потребности он будет удовлетворять. Для этого необходимо предусмотреть существующие решения для выявления достоинств и недостатков.

1.1.1 Анализ актуальности темы

Проблема эффективного взаимодействия между участниками учебного процесса, а именно преподавателя, студента и руководства учебного заведения, является актуальной уже долгое время, если речь идет об организации консультаций.

Прежде всего – это неэффективное информирование о расписаниях консультаций, поскольку они доступны только в бумажном виде на кафедрах, что требует физического присутствия в университете. Бывают ситуации, что нужно узнать расписание на нескольких кафедрах. Соответственно, есть предположение, что нужно посетить не один учебный корпус, значительно теряя свое время. Также нужно учитывать то, что расписания консультаций меняются, особенно часто в начале учебного года. Если преподаватель не сможет провести консультацию, например, в случае болезни, он никак не сможет предупредить об этом студентов, которые нуждаются во встрече.

Второстепенная проблема, которую стоит отметить, – конфликты в неорганизованности очередей студентов, влияющие на их успеваемость. При огромном количестве пришедших на консультацию студентов, что является

					ВКР 090301.06 25 81 01	Лист
Изм.	Лист	№ докум.	Подпись	Дата		10

распространенной ситуацией, возникают очереди, которые создают напряженную обстановку и могут повлиять на качество ее проведения. Поэтому скопление людей может привести к хаотичным очередям и конфликтам между студентами. Наверное, самой типичной ситуацией является то, что студенты, стоящие в конце очереди, могут не успеть посетить консультацию, при этом теряя время и откладывая встречу на следующий раз, которая вновь может повторить данный случай и повлиять на академическую успеваемость.

В-третьих – нагрузка на преподавателей. Проблема пересекается с предыдущей, так как с огромным количеством пришедших студентов преподаватель может не справиться, что может негативно сказаться на качестве проведения консультации, а равномерно распределить нагрузку считается невозможной задачей.

Следующая проблема, являющаяся не менее важной – это отсутствие аналитики для деканата. Руководство факультета, к сожалению, не видит статистики успеваемости студентов, а именно информации по сдаче задолженностей, соответственно, и посещению консультаций. Анализ данных о посещаемости и сдаче задолженностей студентов может служить основой для выявления проблем с успеваемостью или мотивацией. Информация о посещаемости может помочь в эффективном распределении консультаций, чтобы они проводились в удобное время для большинства студентов и преподавателей. Деканат может использовать данные для оценки работы преподавателей, анализируя, насколько результативны их консультации в повышении успеваемости студентов.

Если рассматривать тему глубже, можно выявить еще несколько насущных вопросов, которые будут пересекаться с вышеупомянутыми проблемами. Таким образом, можно сделать вывод, что внедрение системы онлайн-записи студентов на консультации к преподавателям, которое позволит устранить вышеперечисленные сложности, является необходимым решением для коммуникации между всеми участниками образовательного процесса, а именно преподавателем, студентом и руководством.

					ВКР 090301.06 25 81 01	Лист
						11
Изм.	Лист	№ докум.	Подпись	Дата		

1.1.2 Анализ аналогов

Некоторые системы нашли решения в проблеме взаимодействия между преподавателями, студентами и руководством. В рамках обзора предметной области были рассмотрены аналоги в ряде вузов РФ, используемые для организации консультаций.

На сайте МГУТУ им. К. Г. Разумовского [3] студенты имеют возможность просмотреть файл с расширением .xlsx, изображенном на рисунке 1, в котором содержится вся необходимая информация в виде расписания консультаций по приему академических задолженностей, а именно дата и время проведения. Подход к организации консультаций в университете, основанный на статическом графике в виде файла, имеет ряд существенных недостатков.

Наименование факультета	Группа	Курс	Форма обучения	Дисциплина	Мероприятие	Кафедра	ФИО Преподавателя	Период проведения
ФБиРХ	050306-ПРБПв-21/1	4	Очно-заочная форма	Нормирование и снижение загрязнения окружающей среды	Зачет	24 Биологии и биоинформатики	Золотова А.В.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-21/1	4	Очно-заочная форма	Оценка воздействия на окружающую среду	Экзамен	26 Экологии и природопользования	Медякина М.В.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-21/1	4	Очно-заочная форма	Проектирование	Зачет	26 Экологии и природопользования	Попова Е.О.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-21/1	4	Очно-заочная форма	Устойчивое развитие	Экзамен	26 Экологии и природопользования	Перфилов А.А.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-21/1	4	Очно-заочная форма	Экологическая токсикология	Экзамен	26 Экологии и природопользования	Медякина М.В.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-21/1	4	Очно-заочная форма	Экологический менеджмент предприятия	Зачет	26 Экологии и природопользования	Глебова И.А.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-22/1	3	Очно-заочная форма	Геохимия и геофизика биосферы	Экзамен	24 Биологии и биоинформатики	Шкель А.А.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-22/1	3	Очно-заочная форма	Иностранный язык в профессиональной деятельности	Зачет	19 Иностранных языков	Латышева С.Ю.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-22/1	3	Очно-заочная форма	Основы рационального природопользования	Экзамен	26 Экологии и природопользования	Глебова И.А.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-22/1	3	Очно-заочная форма	Безопасность жизнедеятельности	Зачет	42 Пожарной и техносферной безопасности	Макаренко А.И.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-22/1	3	Очно-заочная форма	Геохимия и геофизика биосферы	Экзамен	24 Биологии и биоинформатики	Шкель А.А.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-22/1	3	Очно-заочная форма	Геоэкология	Зачет	26 Экологии и природопользования	Глебова И.А.	17.02.2025-22.03.2025
ФБиРХ	050306-ПРБПв-22/1	3	Очно-заочная форма	Иностранный язык в профессиональной деятельности	Зачет	19 Иностранных языков	Латышева С.Ю.	17.02.2025-22.03.2025

Рисунок 1 – Файл с расписанием академических задолженностей

Конечно же, это решает вопрос физической зависимости от расписания, поскольку студент в онлайн-режиме может просмотреть график. Но такой формат не обеспечивает гибкость и актуальность информации, поскольку студенты, привыкшие к динамическим системам, вынуждены постоянно проверять файл, чтобы узнать о доступных консультациях, чего уж говорить об отмене консультации преподавателем, о которой студент не будет осведомлен. В то же время преподаватель, не имея возможности отслеживать количество студентов,

планировавших посетить консультацию, рискует столкнуться с перегрузкой, что не позволит качественно и эффективно вести прием студентов.

Сайт ЮФУ [4] придерживается такого же традиционного подхода, что и МГУТУ им. К. Г. Разумовского, но при этом имеет возможность преподавателям организовать консультацию, предварительно согласовав со студентом дату и время консультации. В данном решении, представленном на рисунке 2, инициатива организации консультации лежит исключительно на преподавателе, что лишает студента возможности активно участвовать в планировании своего графика обучения. Таким образом, можно сделать вывод о том, что такой подход считается непродуктивным.

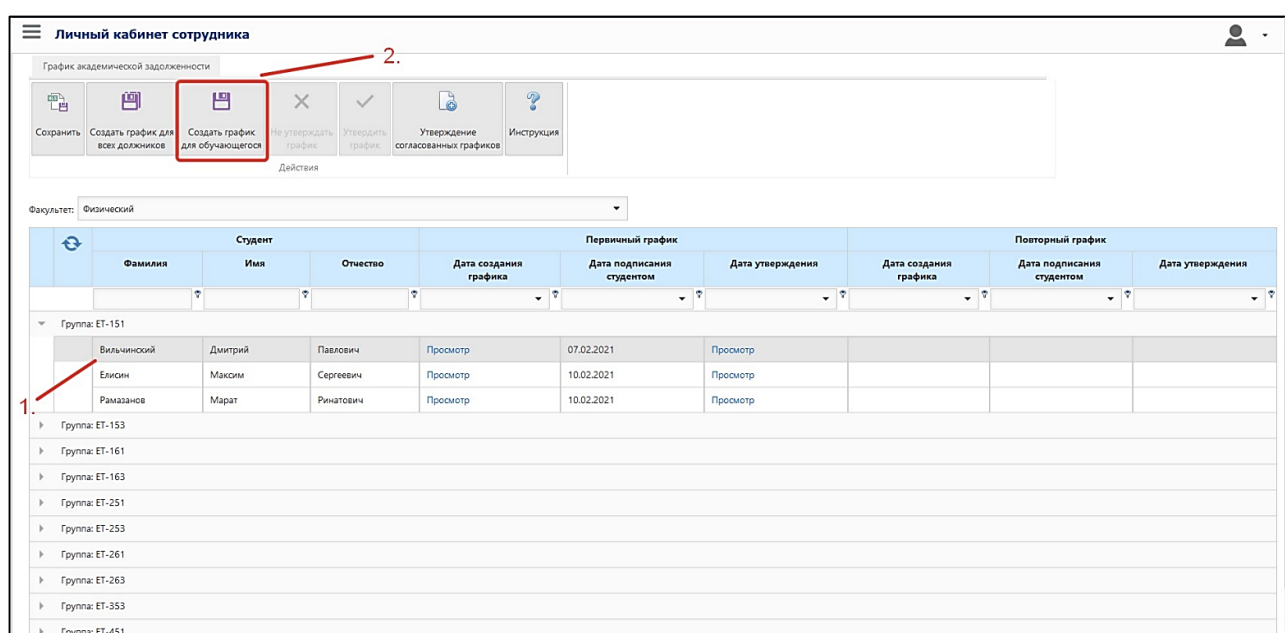


Рисунок 2 – Интерфейс преподавателя для организации консультаций

Рассматривая возможности сайта ПГУ [5], можно отметить, что помимо графика проведения консультации в виде файла и организации консультации преподавателем с подтверждением записи студента имеется возможность создания чата между преподавателем и студентом, представленного на рисунке 3. В нем можно договориться о дате проведения консультации. Но такой подход оказывается менее эффективным, потому что чаты не предназначены для организации консультаций, так как они больше подходят для кратковременного общения. В результате преподаватель может столкнуться с ситуацией, когда ему будет сложно отследить все сообщения, а также согласовывать время и количество

студентов, что приведет к неравномерному распределению нагрузки. К тому же студентам придется тратить время на ожидание ответа от преподавателя, а преподавателю – на отслеживание и обработку множества сообщений. Такой подход к записи на консультацию считается неэффективным.

Главная » Портфолио

Портфолио. Дарья Александровна Нагорная

Файлы студента Подключенные устройства **Чат**

ID ЭИОС: 517710386

Сведения о вакцинации: Сертификат о вакцинации 02.09.2021 - 02.09.2022 Изменить

Табельный/регистрационный номер: 72482

Дата рождения: 19.06.2003, гор. Пенза

Редактировать биографию

Не трудоустроен через ПС "ВУЗ+Работодатель"

Телефон: 8(937) 913-67-99 Изменить

E-mail: nda03@yandex.ru Изменить

Тип файла: ".jpg"

Загрузить фото

Резюме

ВУЗ + РАБОТОДАТЕЛЬ

Добавить ссылку на профиль социальной сети

Показывать дополнительную информацию в личном кабинете

Не показывать профиль неавторизованному пользователю

Рисунок 3 – Интерфейс студента с возможностью создания чата

Анализируя все предложенные решения можно сделать вывод, что не все проблемы, выявленные ранее, можно считать решенными. Достоинства и недостатки представлены в таблице 1.

Таким образом, система МГУТУ им. К. Г. Разумовского абсолютно не решает ни одной ключевой проблемы из-за полного отсутствия автоматизации, так как является лишь электронным аналогом бумажного расписания. В решении ЮФУ ситуация лучше, так как помимо статического файла с расписанием академических задолженностей смогла внедрить систему организации консультации преподавателем, частично решив проблему с очередями, но, к сожалению, без автоматической записи и аналитики. Система ПГУ, которая предлагает и статический файл с графиком консультации, и организацию консультации преподавателем с подтверждением записи студентом, недавно ввела коммуникацию между студентом и преподавателем через чат (средство обмена сообщениями в режиме реального времени), который, как уже было выявлено

ранее, не является эффективным решением.

Таблица 1 – Сравнительная таблица аналогов и разрабатываемой системы

Критерий	Веб-системы			
	МГУТУ им. К. Г. Разумовского	ЮФУ	ПГУ	Разрабатываемая система
Запись на консультацию	—	+	+	+
Контроль нагрузки преподавателя	—	+	+	+
Минимизация живых очередей	—	+	+	+
Фиксация результатов консультаций	—	—	+	+
Аналитика для деканата	—	—	+	+
Гибкость создания консультаций	—	—	—	+
Эффективное информирование о расписании консультаций	—	—	—	+
Уведомления об изменениях	—	—	—	+

Разрабатываемая система онлайн-записи позволит решить все описанные проблемы для улучшения взаимодействия между преподавателем, студентом и руководством учебного подразделения. Для этого ниже будет приведен анализ технического задания.

1.2 Анализ технического задания

Анализ технического задания связан с формулировкой задачи и описанием требований к процессу создания ПО. Здесь определяются основные возможности и функциональные требования к реализации будущей системы.

1.2.1 Постановка задачи

При проведении анализа предметной области можно составить следующую задачу: реализовать веб-приложение как подсистему ЭИОС, автоматизирующее процесс самостоятельной записи студентов на консультации, которые организует преподаватель лично, и предоставляющее инструмент для аналитики

по успеваемости деканату, а именно данные по посещаемости консультаций и сдаче задолженностей в виде диаграмм.

1.2.2 Анализ требований к серверной и клиентской части на разработку

Конечным этапом анализа предметной области и технического задания является анализ требований на разработку для удовлетворения потребностей пользователей.

Основные требования к серверной части:

- центральный процессор с двумя ядрами и тактовой частотой от 2 ГГц;
- оперативная память объемом от 4 ГБ;
- жесткий диск от 10 ГБ свободного пространства;
- доступ в интернет или локальную сеть;
- осуществление аутентификации и авторизации пользователя с использованием уже существующей базы данных с разграничением доступа в зависимости от ролевой модели: преподаватель, студент, руководство;
 - создание у преподавателя регулярных (с указанием дня недели и возможностью создания консультации по первой или второй неделе, а также даты и времени удаления) и разовых (с указанием конкретной даты) консультаций с настройками: время начала и время окончания консультации, аудитория и максимальное количество студентов;
 - предоставление преподавателю данных о регулярных и разовых консультациях с возможностью их удаления;
 - предоставление преподавателю списка записавшихся студентов на консультацию с возможностью удаления записи студента;
 - предоставление преподавателю архива записей студентов с возможностью редактирования данных о посещаемости, сдаче задолженностей студентов, оставления отзыва, и указанием скрытия архивной записи;
 - предоставление студенту списка доступных консультаций;
 - запись студента на консультацию к преподавателю с указанием причины записи;
 - удаление записи студентом;

- предоставление студенту его архива записей;
- отправка оповещений (уведомлений) преподавателям и студентам в случае изменения данных о консультациях и записях с возможностью их прочтения;
- предоставление руководству архива записей и аналитики посещаемости и сдачи задолженностей в виде диаграмм и подписей с процентами.

Основные требования к клиентской части:

- интуитивно понятный и адаптивный интерфейс для обеспечения удобства использования и доступности веб-приложения на разных устройствах: смартфонах, планшетах, настольных компьютеров и других гаджетов;
- динамическое обновление данных без перезагрузки страницы, что делает веб-приложение быстрым и отзывчивым;
- вход пользователя на соответствующую страницу в зависимости от роли через форму авторизации;
- создание, чтение и удаление всех типов консультаций, предоставляемые преподавателю;
- удаление записей студентов с указанием причины удаления, предоставляемое преподавателю;
- редактирование данных о посещаемости, сдаче задолженностей студентов, оставления отзыва и возможность скрытия архивной записи, предоставляемое преподавателю;
- создание, чтение и удаление записи, предоставляемое студенту;
- чтение архивных записей, предоставляемое студенту;
- прочтение оповещений (уведомлений), предоставляемое преподавателям и студентам;
- чтение архива записей студентов на консультации к преподавателям и аналитики посещаемости и сдачи задолженностей в виде диаграмм и подписей с процентами, предоставляемой руководству.

Теперь, когда было определено, что должно делать приложение, можно переходить к проектированию ПО.

2 Проектирование программного обеспечения

Вторым этапом разработки ПО является проектирование, в котором определяется выбор технологического стека, архитектура системы, описание структуры базы данных, компоненты и поведение системы.

2.1 Выбор технологий и инструментов разработки

В данном подразделе рассмотрены технологии и инструменты, которые были выбраны для разработки клиент-серверного приложения. Здесь необходимо учитывать большинство критериев: производительность, поддержка, совместимость, безопасность, гибкость, расширяемость, скорость разработки и многое другое.

2.1.1 Выбор языка программирования для серверной части

Существует большое количество языков программирования для реализации серверной части веб-приложений. Согласно исследованию [6], среди топ-3 языков программирования в 2025 году для серверной разработки лидируют Python, Java и C#, особенности которых будут приведены в таблице 2.

Таблица 2 – Сравнение языков программирования для серверной части

ЯП	Достоинства	Недостатки
Python	– простота и лаконичность кода – быстрая разработка – большое количество библиотек и фреймворков для веб-разработки	– меньшая производительность – не подходит для высоконагруженных приложений – меньше возможностей для масштабируемой многопоточности
Java	– платформенная независимость – большое количество библиотек и фреймворков для веб-разработки – надежность и безопасность – больше возможностей для масштабируемой многопоточности	– более сложный синтаксис – большой объем кода для реализации задач – склонность к более высокому потреблению памяти
C#	– интеграция с продуктами Microsoft – поддержка асинхронного программирования	– платформенная зависимость – меньше библиотек и фреймворков для веб-разработки – ограниченная поддержка на других ОС

Разрабатываемое веб-приложение считается подсистемой ЭИОС университета, поэтому нужно выделить важность интеграции с существующими

системами, с которой у С# могут возникнуть проблемы. В системе используется разграничение доступа по типам пользователей, поэтому нужно учитывать поддержку ролей и авторизации, а с этим хорошо справляются Python и Java. Абсолютно точно нужно брать в расчет масштабируемость и многопоточность, с чем плохо справляется Python в отличие от других ЯП, поскольку данная система со временем может расти из-за увеличения числа пользователей, консультаций, записей, аналитики. Для университета важна защита данных у всех участников системы, поэтому нужно ориентироваться на создание надежных и безопасных приложений. Еще важна высокая производительность под нагрузкой, с чем хуже справляется Python по сравнению с другими рассматриваемыми ЯП, поскольку является интерпретируемым языком программирования и работает медленнее Java. Python также является слабо типизированным языком программирования, то есть нет строгой проверки типов, в отличие от Java, который отловит проблему на этапе компиляции. Поскольку продуктом является веб-приложение, у С#, у которого лучшая поддержка – под Windows, меньше готовых решений для веб-разработки.

Таким образом, исходя из вышеперечисленных требований, можно сделать вывод, что выбор строго типизированного объектно-ориентированного языка программирования общего назначения Java является лучшим решением для разработки серверной части данной системы.

2.1.2 Выбор языка программирования для клиентской части

Однозначным ответом в выборе технологии для клиентской части веб-приложения является JavaScript, так как этот язык является стандартом и альтернатив практически нет.

Данный язык программирования поддерживается всеми современными браузерами, что делает его универсальным. JavaScript позволяет создавать интерактивные и динамичные веб-страницы и легко работать с асинхронными запросами без полной перезагрузки страницы, что будут использоваться в данной системе.

Помимо JavaScript, будут использоваться язык гипертекстовой разметки

HTML, который описывает структуру документа, и язык стилей CSS, описывающий внешний вид страниц и разделяющий контент и оформление.

2.1.3 Выбор фреймворка и его модулей

Фреймворки предоставляют готовые решения для многих видов задач, что позволяют значительно сократить объем программного кода, а также увеличить его поддерживаемость, читаемость и понятность. Среди современных фреймворков, предназначенных для упрощения разработки приложений на языке Java, можно выделить Spring Framework, Jakarta EE и Micronaut, сравнение которых представлено в таблице 3.

Таблица 3 – Сравнение фреймворков

Критерий	Spring Framework	Jakarta EE	Micronaut
Разработка	Быстрая, автоматическая настройка	Медленная, больше ручной настройки	Быстрая, автоматическая настройка
Интеграция с внешними системами	Отличная, большое количество библиотек	Средняя, зависит от реализации сервера приложений	Средняя, меньше готовых решений
Масштабируемость	Высокая, подходит для крупных приложений	Средняя	Высокая, подходит для микросервисов
Сообщество	Огромное, большое количество документации	Среднее, меньше новых проектов	Растущее

Подходящим фреймворком для разработки является Spring Framework, поскольку не требует ручной настройки, содержит множество библиотек, оптимизирован для крупных приложений и имеет множество готовых решений. Но выбора фреймворка недостаточно, поскольку необходимы его модули для решения конкретных задач.

Для быстрого запуска, автоматической настройки зависимостей и компонентов, а также сокращения времени на ручную конфигурацию был использован модуль Spring Boot [7]. К тому же основным преимуществом применения данного модуля является задействование встроенного сервера Tomcat без необходимости самостоятельной настройки отдельного сервера.

В данной системе используется механизм аутентификации и авторизации пользователей, с чем сможет справиться Spring Security [8], так как обеспечивает

безопасность приложений. Поскольку в программном продукте используется ролевая модель, будет легко реализовать переход пользователя на разные страницы именно с этим модулем. Так же данный набор инструментов будет решать вопросы с управлением сессии и настройкой ее время истечения. К тому же в БД будут храниться хешированные пароли пользователей, а данный модуль предоставляет интерфейсы для безопасного кодирования паролей.

Модуль Spring Data JPA представляет мощный инструмент для упрощения работы с БД. Преимуществом является уменьшение шаблонного кода, поскольку есть возможность создавать интерфейсы-репозитории, а Spring позволит автоматически сгенерировать реализацию базовых методов, что сэкономит время и уменьшит количество ошибок. Кроме того, Spring Data JPA может автоматически создавать запросы на основе имен методов в репозитории.

Последний модуль, который будет использоваться в системе – Spring MVC, применяемый для создания веб-приложения с использованием архитектуры MVC, о котором подробно будет говориться позже. Он делит систему на три компонента:

- модель, отвечающий за управление и бизнес-логику приложения;
- представление, отвечающее за отображение информации пользователю;
- контроллер, обеспечивающий взаимодействие между моделью и представлением.

2.1.4 Выбор базы данных

Главный вопрос этого подраздела – выбор БД по структуре организации данных. Для разрабатываемой системы больше всего подходит реляционная база данных.

Важным критерием разработки веб-приложения онлайн-записи являются масштабируемость и производительность системы. РБД как раз могут обрабатывать большие объемы данных, что подходит для университета с большим количеством человек. Преимуществом РБД является структурированное хранение данных, поскольку позволяет организовать данные в таблицах с четко определенными связями между ними и ограничениями, что обеспечивают согласованность и надежность данных.

РБД обладают простотой использования, поскольку использует язык структурированных запросов SQL для манипуляции различными данными, который не является самостоятельным языком программирования, но используется для взаимодействия с базой данных через СУБД.

Важным свойством РБД является согласованность и целостность данных. Если во время выполнения транзакции произойдет ошибка, то все изменения, сделанные до этого момента, будут отменены. Соответственно, данные останутся в правильном виде.

Для сравнительного анализа, визуальной представленной в таблице 4, были выбраны следующие СУБД, которые соответствуют рассматриваемой структуре организации данных: PostgreSQL, MySQL, SQLite.

Таблица 4 – Сравнение СУБД

Критерии	PostgreSQL	MySQL	SQLite
Лицензия	PostgreSQL Licence (открытая)	GPL (открытая), Commercial	Public Domain (открытая)
Поддержка SQL	Полная	Полная	Ограниченная
Поддержка JSON	Полная	Ограниченная	Минимальная
Совместимость с Java	Отличная	Отличная	Базовая
Настройка	Сложная	Средняя	Минимальная
Производительность	Высокая для сложных запросов и больших нагрузок	Высокая для простых запросов, низкая для аналитических запросов	Низкая для сложных запросов
Масштабируемость	Высокая	Высокая	Низкая
Расширяемость	Высокая	Средняя	Низкая
Инструменты	Широкий выбор	Широкий выбор	Ограниченный выбор
Использование	Универсально	Веб-приложения	Встраиваемые приложения
Безопасность	Высокая	Хорошая	Базовая

Очевидным выбором является PostgreSQL [9], хорошо подходящим для разработки веб-приложения. Несмотря на сложность настройки, гибкость которого

требует изучения, данная СУБД имеет открытую лицензию, являющаяся важным критерием, поскольку обеспечивает свободное использование, высокую производительность, которая характеризуется современной многопоточной архитектурой, что позволяет эффективно обрабатывать запросы и параллельно выполнять операции. Также она обеспечивает различные методы масштабируемости, разрешающая обрабатывать большие объемы данных, содержит большой выбор инструментов, позволяя добавлять собственные типы данных, функции, агрегаты, операторы и индексы.

2.1.5 Выбор графического интерфейса для управления базой данных

На самом деле, в большинстве случаев, для работы с PostgreSQL выбирают графический интерфейс pgAdmin 4, поскольку инструмент разработан сообществом СУБД PostgreSQL и является официальным средством для администрирования. Но тем не менее необходимо рассмотреть аналоги и составить сравнительный анализ, продемонстрированный в таблице 5.

Таблица 5 – Сравнение инструментов с графическим интерфейсом

Критерий	pgAdmin 4	DBeaver	TablePlus
Цена	Бесплатно	Бесплатно	Платно
Поддержка SQL	Полная	Базовая	Базовая
Поддержка PostgreSQL	Нативная интеграция	Частичная	Базовая
Функциональность	Полная	Хорошая	Упрощенная
Производительность	Стабильная	Быстрая	Очень быстрая
Визуализация данных	Да	Да	Минимальная
Расширения	Да	Ограниченная поддержка	Нет

Лучшим выбором для профессиональной работы с PostgreSQL считается графический инструмент pgAdmin, являющийся официальным графическим инструментом для управления базами данных PostgreSQL, который гарантирует 100% совместимость и доступ ко всем функциям. Данная утилита обладает интуитивно понятным интерфейсом, подходящим к изучению даже для начинающих программистов, а также включает мониторинг и диагностику производительности и состояния баз данных в реальном времени, что

является немаловажным преимуществом.

2.1.6 Выбор среды разработки

Выбор интегрированной среды разработки влияет на разные аспекты процесса разработки ПО.

Программный комплекс с обширными функциональными возможностями, объединяющий в себе инструменты для реализации приложений, как правило, состоит из текстового редактора для написания кода, компилятора, отладчика и инструментов для автоматизации сборки кода. Исследование и детальный сравнительный анализ между такими IDE, как IntelliJ IDEA, Eclipse и NetBeans, рассмотрен в таблице 6.

Таблица 6 – Сравнение интегрированных сред разработки

Критерий	IntelliJ IDEA	Eclipse	NetBeans
Лицензия	Community (бесплатная), Ultimate (платная)	Open Source (бесплатная)	Apache License (бесплатная)
Настройка	Минимальная	Сложная	Средняя
Поддержка Java	Отличная	Отличная	Отличная
Интеграция с БД	Имеются встроенные инструменты	Требуется установка плагинов	Имеется простой SQL-редактор
Производительность	Быстрая	Медленная	Средняя
Расширяемость	Гибкая	Модульная	Адаптивная
Веб-разработка	Встроенная	Требуется установка плагинов	Базовая
Текстовый редактор	Продвинутый	Базовый	Базовый
Компиляция	Встроенная	Встроенная	Встроенная
Отладка	Полноценная	Полноценная	Полноценная
Сборка	Встроенная	Встроенная	Встроенная

Анализируя таблицу, можно сделать вывод, что более разумным выбором будет IntelliJ IDEA.

Интегрированная среда разработки, доступная в открытой версии и подходящая для личного и коммерческого использования, имеет широкий набор инструментов.

Основными компонентами являются:

- текстовый редактор с подсветкой синтаксиса, поддерживающий большинство языков программирования, автодополнение кода и настройку стиля кода;
- компилятор и интерпретатор – программы, которые отвечают за преобразование кода;
- отладчик, позволяющий останавливать выполнение в определенных точках останова и анализировать состояние приложения;
- система сборки (используется Maven), необходимая для получения информационного продукта из исходного кода.

Выбор IntelliJ IDEA подходит для удобной работы с БД, так как поддерживает клиентскую и серверную часть разработки приложений, обладает автоматическим развертыванием на встроенном веб-сервере Tomcat и большим набором профессиональных инструментов.

2.2 Архитектура клиент-серверного приложения

Как уже было сказано ранее, веб-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Все веб-приложения задействованы на клиент-серверной архитектуре [10], проиллюстрированной на рисунке 4, с разделением на два компонента:

а) Клиент – это часть системы, работающая на устройстве пользователя (веб-браузере, мобильном или десктопном приложении), отвечающая за отображение интерфейса и взаимодействие с пользователем и позволяющая запрашивать необходимую информацию у сервера, отправляя запросы для получения необходимых данных.

б) Сервер – это часть системы, работающая на удаленном компьютере, которая обрабатывает запросы и предоставляет запрашиваемую информацию клиенту.

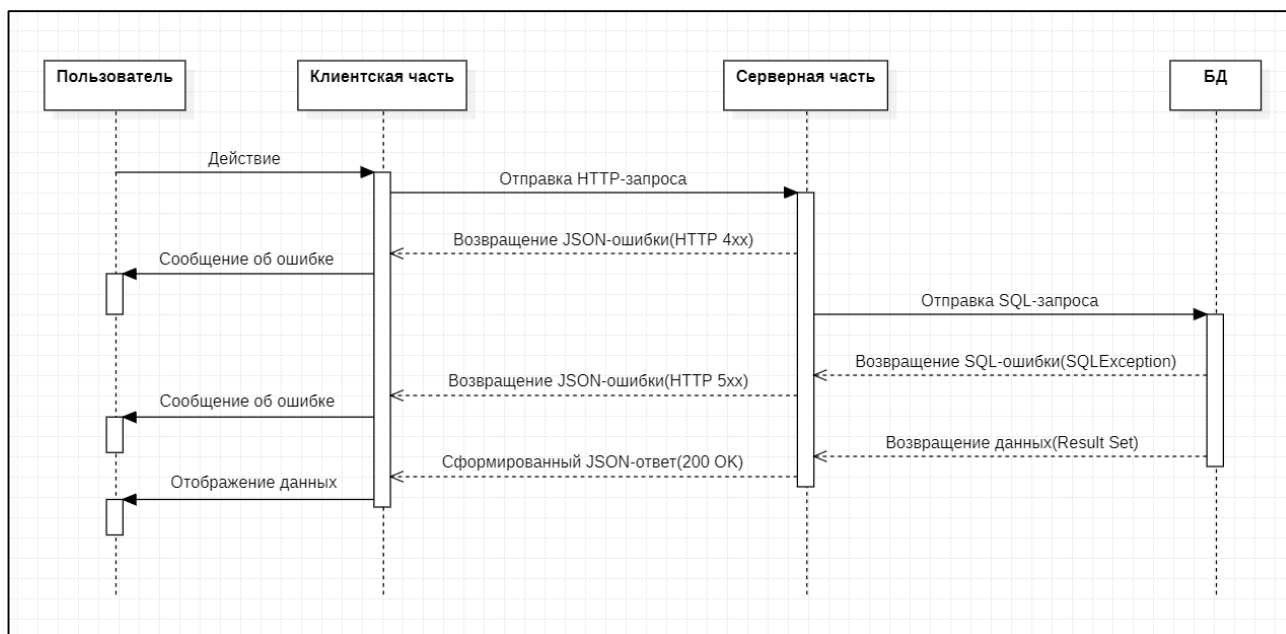


Рисунок 4 – UML-диаграмма последовательности взаимодействия компонентов веб-приложения

Рассматривая рисунок, который представляет взаимодействие между пользователем, клиентской частью, серверной частью и базой данных, необходимо проанализировать следующие сценарии:

а) Обработка успешного запроса. Пользователь выполняет действие в интерфейсе, взаимодействуя с клиентской частью системы, что приводит к формированию и отправке HTTP-запроса (GET, POST, PUT, DELETE) к серверу. Сервер принимает запрашиваемый запрос и, проверяя валидность, формирует SQL-запрос (SELECT, INSERT, UPDATE, DELETE) к базе данных в случае корректности запроса. База данных, в свою очередь, обрабатывает SQL-запрос и возвращает успешный результат (ResultSet) серверу. Возвращаемся к серверной части, которая преобразует данные из БД в JSON-ответ с кодом 200 OK, и затем клиентской части, которая получает JSON-ответ и отображает данные в интерфейсе, доступный пользователю, который видит запрашиваемые данные.

б) Обработка HTTP-запроса с ошибкой на стороне клиента. Это довольно частая ситуация, когда ошибка связана с некорректными действиями пользователя или клиента: неправильный синтаксис запроса (400 Bad Request), отсутствие аутентификации (401 Unauthorized), распознавание и невыполнение запроса из-за отсутствия аутентификации (403 Forbidden) и другие. Если HTTP-запрос не

прошел проверку или произошла ошибка, то серверная часть возвращает HTTP-ответ с кодом ошибки 4xx (JSON-ошибкой).

в) Обработка SQL-запроса с ошибкой на стороне сервера. Данная ситуация встречается реже, но считается намного критичнее, когда ошибка возникает из-за проблем на сервере или в БД, с которой не может справиться клиент, и требует вмешательства разработчика: неожиданная ошибка (500 Internal Server Error), отсутствие поддержки или невозможности обработки типа отправленного запроса (502 Bad Implemented), получение некорректного ответа от другого сервера от шлюза или промежуточного сервера (503 Bad Gateway) и другие. Если база данных сталкивается с проблемой, то возвращается SQLException, а серверная часть ловит исключение и формирует HTTP-ответ с кодом ошибки 5xx (JSON-ошибкой).

2.3 Проектирование базы данных

Данный подраздел описывает наиболее важную стадию на этапе проектирования ПО. Необходимо определить, какие данные будут храниться в БД и как они будут использоваться. Помимо этого, важно показать, как объекты реального мира, представляющие определенные свойства, будут взаимодействовать друг с другом. Поэтому речь пойдет о проектировании БД, без которого будет приходиться трудно на этапе разработки ПО следующим причинам:

а) постоянное изменение структуры БД, так как необходимо будет часто менять схему, что приводит к несогласованности;

б) ошибки в данных и связях, которые могут привести к дублированию, потере целостности и некорректным запросам и данным;

в) проблемы с масштабированием, потому что без ясного проектирования сложно предугадать, как база данных будет развиваться, что может привести к проблемам при добавлении новых функций, если изначальная структура заложена гибко.

Поэтому ниже будет рассмотрен вопрос о структуре БД, которая поможет определить, какие таблицы необходимы, как будут организованы данные и как будут взаимодействовать между собой сущности.

2.3.1 Инфологическое проектирование

Инфологическое или концептуальное проектирование нацелено на создание инфологической модели, приведенной на рисунке 5, которая не зависит от конкретных технологий и СУБД, поскольку ориентировано на семантику предметной области. Она лишь описывает сущности и взаимосвязи и является основой проектируемой базы данных.

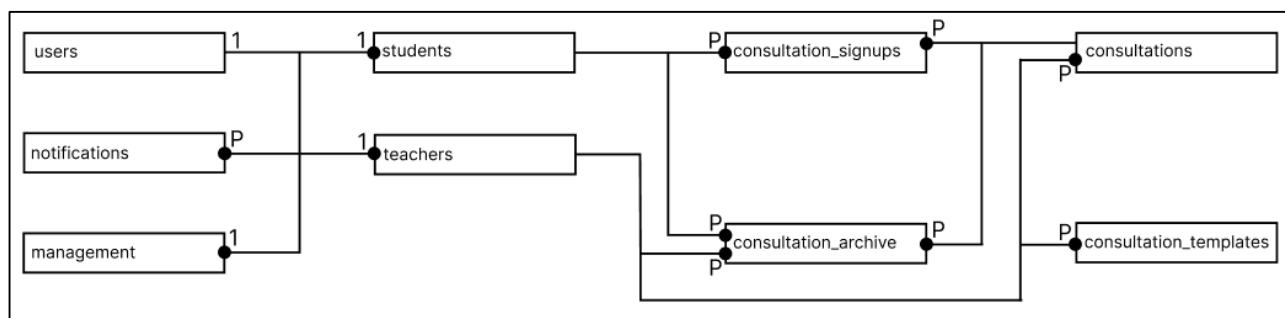


Рисунок 5 – Инфологическая модель данных

Данная модель служит образом реального мира и определяет основные сущности и их связи (в данной модели используются только два типа связи) без конкретной детализации. В таблице 7 будет рассмотрено подробное описание связей между сущностями системы. Для данной системы предложены следующие таблицы:

- users – содержит общую информацию о пользователе для их аутентификации и авторизации;
- teachers – предназначена для хранения общей информации о преподавателях;
- students – предназначена для хранения общей информации о студентах;
- management – предназначена для хранения общей информации о руководстве;
- notifications – фиксирует уведомления, отправленные пользователям;
- consultation_templates – используется для хранения шаблона регулярной консультации;
- consultations – содержит информацию о конкретной консультации;
- consultation_signups – хранит данные записей студентов на консультации;
- consultation_archive – содержит информацию о архиве записей студентов.

Таблица 7 – Описание таблиц и связей

Название таблицы	Связь с таблицей	Описание связи	Тип связи
users	students	Наследование пользователей по роли студентов	Один ко одному, идентифицирующая (1:1)
	teachers	Наследование пользователей по роли преподавателей	Один ко одному, идентифицирующая (1:1)
	management	Наследование пользователей по роли руководства	Один ко одному, идентифицирующая (1:1)
	notifications	Получение уведомлений всем пользователями	Один ко многим, идентифицирующая (1:N)
teachers	consultation_templates	Создание множества регулярных консультаций преподавателем	Один ко многим, идентифицирующая (1:N)
	consultations	Привязка к преподавателю каждой конкретной консультации для ее управления	Один ко многим, идентифицирующая (1:N)
	consultation_archive	Привязка к преподавателю каждого архива записи студента для ее управления и учета	Один ко многим, идентифицирующая (1:N)
consultation_templates	consultations	Построение конкретных консультаций по регулярному шаблону	Один ко многим, идентифицирующая (1:N)
consultations	consultation_signups	Записи студентов на конкретные консультации	Один ко многим, идентифицирующая (1:N)
	consultation_archive	Архив записей студентов на конкретные консультации	Один ко многим, идентифицирующая (1:N)
students	consultation_signups	Привязка записи к студенту для ее управления и учета	Один ко многим, идентифицирующая (1:N)
	consultation_archive	Привязка записи к студенту для ее учета	Один ко многим, идентифицирующая (1:N)

Взаимодействие таблиц обеспечено идентифицирующей связью. В разрабатываемой системе будут использоваться только два типа связи: один к одному и один ко многим.

2.3.2 Логическое проектирование

Логическое или датологическое проектирование построено на создании схемы БД, иллюстрированной на рисунке 6, на основе конкретной модели данных, а именно реляционной. Здесь уже разворачивается инфологическая модель, добавляя при этом описание атрибутов и определяя набор отношений с указанием первичных и внешних ключей для их связи.

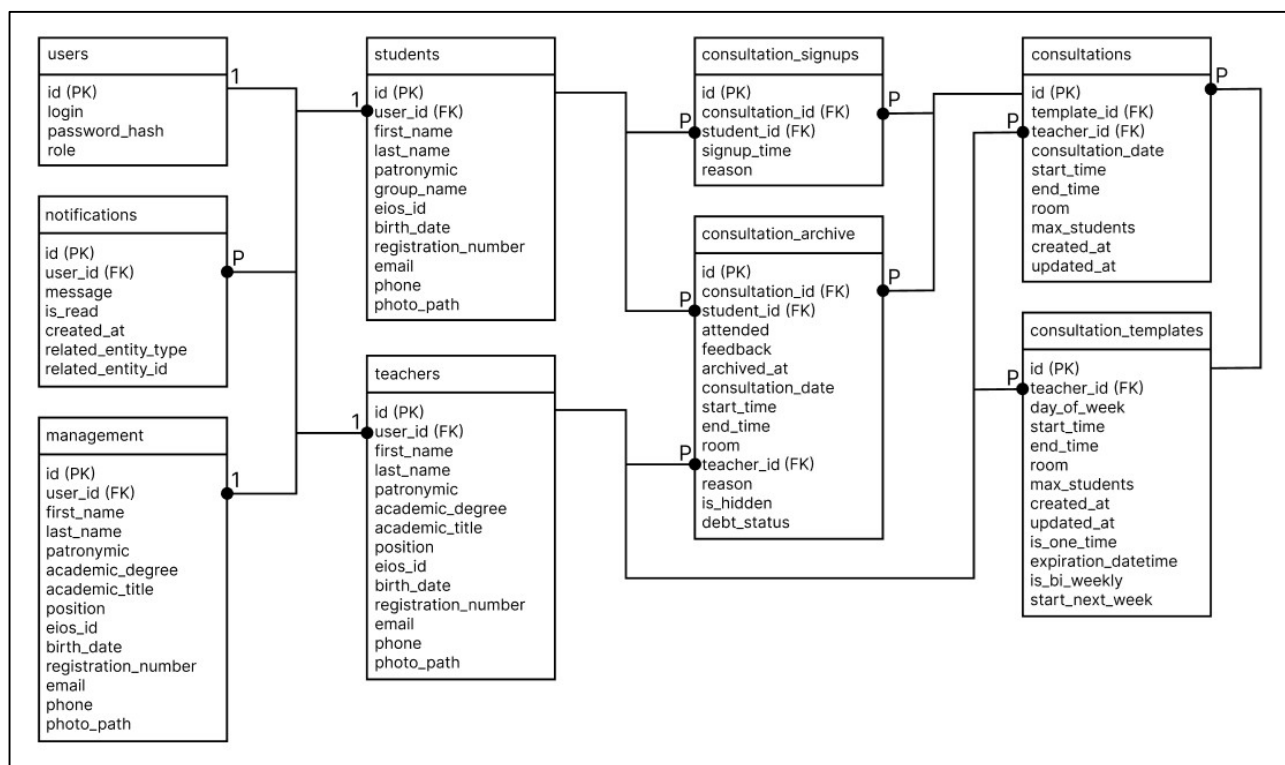


Рисунок 6 – Логическая модель данных

В таблице 8 приведено описание сущности пользователя и его атрибутов, которая представляет базовое отношение наследования пользователей по ролям.

Таблица 8 – Атрибуты таблицы users

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор пользователя	PK
login	Логин пользователя	
password_hash	Хешированный пароль пользователя	
role	Роль пользователя	

В таблице 9, 10 и 11 описаны сущности участников систем и описание их атрибутов.

Таблица 9 – Атрибуты таблицы teachers

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор преподавателя	РК
user_id	Идентификатор пользователя	FK
first_name	Фамилия преподавателя	
last_name	Имя преподавателя	
patronymic	Отчество преподавателя	
academic_degree	Ученая степень преподавателя	
academic_title	Ученое звание преподавателя	
position	Должность преподавателя	
eios_id	Идентификатор в ЭИОС преподавателя	
birth_date	Дата рождения преподавателя	
registration_number	Табельный/регистрационный номер преподавателя	
email	Почта преподавателя	
phone	Телефон преподавателя	
photo_path	Фотография преподавателя	

Таблица 10 – Атрибуты таблицы students

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор студента	РК
user_id	Идентификатор пользователя	FK
first_name	Фамилия студента	
last_name	Имя студента	
patronymic	Отчество студента	
group_name	Группа студента	
eios_id	Идентификатор в ЭИОС студента	
birth_date	Дата рождения студента	
registration_number	Табельный/регистрационный номер студента	
email	Почта студента	
phone	Телефон студента	
photo_path	Фотография студента	

Названия полей таблиц преподавателя и руководства будут совпадать.

Таблица 11 – Атрибуты таблицы management

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор руководства	РК
user_id	Идентификатор пользователя	FK
first_name	Фамилия руководства	
last_name	Имя руководства	
patronymic	Отчество руководства	
academic_degree	Ученая степень руководства	
academic_title	Ученое звание руководства	
position	Должность руководства	
eios_id	Идентификатор в ЭИОС руководства	
birth_date	Дата рождения руководства	
registration_number	Табельный/регистрационный номер руководства	
email	Почта руководства	
phone	Телефон руководства	
photo_path	Фотография руководства	

Для оповещения студентов и преподавателей о возможных изменениях в организации консультаций и записей понадобится сущность, которая продемонстрирована в таблице 12, предназначенная для отправки уведомлений данным типам пользователей.

Таблица 12 – Атрибуты таблицы notifications

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор уведомлений	РК
user_id	Идентификатор пользователя	FK
message	Сообщение уведомления	
is_read	Прочтение сообщения уведомления	
created_at	Создание уведомления	
related_entity_type	Тип связанной сущности	
related_entity_id	Идентификатор связанной сущности	

Таблицы 13 и 14 предназначены непосредственно для организации консультаций преподавателем.

Таблица 13 – Атрибуты таблицы consultation_templates

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор шаблона регулярной консультации	PK
teacher_id	Идентификатор преподавателя	FK
day_of_week	День недели регулярной консультации	
start_time	Время начала регулярной консультации	
end_time	Время окончания регулярной консультации	
room	Аудитория, где будет проходить консультация	
max_students	Максимально количество студентов	
created_at	Время создания поля	
updated_at	Время изменения поля	
is_one_time	Разделение двух типов консультаций	
expiration_datetime	Дата удаления консультации	
is_bi_weekly	Создание консультации через неделю	
start_next_week	Создание шаблона со следующей недели	

Таблица 14 – Атрибуты таблицы consultations

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор конкретной консультации	PK
template_id	Идентификатор шаблона регулярной консультации	FK
teacher_id	Идентификатор преподавателя	FK
consultation_date	Дата конкретной консультации	
start_time	Время начала конкретной консультации	
end_time	Время окончания конкретной консультации	
room	Аудитория, где будет проходить консультация	
max_students	Максимально количество студентов	
created_at	Дата и время создания поля	
updated_at	Дата и время изменения поля	

Для записей студентов на консультации к преподавателем и их учета понадобятся две сущности, представленные в таблицах 15 и 16. Данные сущности фиксируют записи студентов, связанные с конкретными консультациями, а так же хранят информацию о посещении, отзыве и статусе задолженностей, которые будет отмечать преподаватель для того, чтобы руководство смогло отслеживать успеваемость.

Таблица 15 – Атрибуты таблицы consultation_signups

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор записи студента	PK
consultation_id	Идентификатор конкретной консультации	FK
student_id	Идентификатор студента	FK
signup_time	Дата и время создания поля	
reason	Причина записи студента	

Таблица 16 – Атрибуты таблицы consultation_archive

Название атрибута	Описание атрибута	Роль атрибута
id	Идентификатор архивной записи студента	PK
consultation_id	Идентификатор конкретной консультации	FK
student_id	Идентификатор студента	FK
attended	Посещаемость студента	
feedback	Отзыв преподавателя	
archived_at	Дата и время создания поля	
consultation_date	Дата конкретной консультации	
start_time	Время начала конкретной консультации	
end_time	Время окончания конкретной консультации	
room	Аудитория, где будет проходить консультация	
teacher_id	Идентификатор преподавателя	FK
reason	Причина записи студента	
is_hidden	Скрытие архивной записи	
debt_status	Статус сдачи задолженности	

Представленные таблицы демонстрируют мощную структуру для взаимодействия участников систем через систему онлайн-записи студентов на консультации к преподавателям и отслеживания успеваемости руководством для повышения качества учебного процесса.

2.3.3 Физическое проектирование

Создание схемы базы данных из логической модели для конкретной СУБД проходит на этапе физического проектирования, поскольку учитывает ее особенности, включая ограничения на именование и типы данных. Без этого этапа невозможно будет реализовать эффективную, надежную и безопасную систему.

Представленные в логическом проектировании сущности будут раскрывать тип данных атрибутов, что важно при реализации ПО, и ограничения, которые, для удобства, будут представлены на этапе разработки программного продукта. Правильный выбор типов данных важен для эффективного и надежного хранения и обработки информации, а также для оптимизации, поскольку есть возможность сократить место и повысить доступность данных, и производительности, так как это повлияет на повышение доступности данных.

В таблицах 17-24 будут представлены типы связей и примеры данных в атрибутах каждой сущности.

Таблица 17 – Атрибуты таблицы users

Название атрибута	Тип данных	Пример данных
id	integer	1
login	character varying(50)	"student1"
password_hash	character varying(255)	"\$2a\$10\$пАOtVpN2nvedgXlVC8yWN.ZmSEcRU1AQMg9GmbwgBEbimV4cfIxOa"
role	character varying(20)	"Студент"

Таблица 18 – Атрибуты таблицы notifications

Название атрибута	Тип данных	Пример данных
id	integer	180
user_id	integer	1
message	text	"Ваша запись на консультацию с преподавателем Кузнецов Дмитрий Владимирович на 2025-05-27 была отменена преподавателем. Причина: Просьба записаться на следующую неделю, так как на сегодняшний день принимаю курсовые работы у студентов"
is_read	boolean	false
created_at	timestamp without time zone	"2025-05-22 00:30:25.238665"
related_entity_type	character varying(50)	"CONSULTATION"
related_entity_id	integer	155

Поскольку в таблицах преподавателя и руководства одинаковые поля,

полностью повторяющие друг друга, логичнее будет их объединить.

Таблица 19 – Атрибуты таблицы teachers и management

Название атрибута	Тип данных	Пример данных
id	integer	1
user_id	integer	4
first_name	character varying(100)	"Дмитрий"
last_name	character varying(100)	"Кузнецов"
patronymic	character varying(100)	"Владимирович"
academic_degree	character varying(50)	"Кандидат наук"
academic_title	character varying(50)	"Профессор"
position	character varying(50)	"Заведующий кафедрой"
eios_id	character varying(20)	"517710387"
birth_date	date	"1981-04-28"
registration_number	character varying(20)	"12482"
email	character varying(100)	"dmitryKux28@yandex.com"
phone	character varying(20)	"82656665415"
photo_path	character varying(255)	"cbaae835-6a03-4fe3-8a9e-d978ffdc783d.jpg"

Таблица 20 – Атрибуты таблицы students

Название атрибута	Тип данных	Пример данных
id	integer	1
user_id	integer	1
first_name	character varying(100)	"Сергей"
last_name	character varying(100)	"Сорокин"
patronymic	character varying(100)	"Петрович"
group_name	character varying(20)	"22BBC1"
eios_id	character varying(20)	"543202659"
birth_date	date	"2003-01-24"
registration_number	character varying(20)	"65776"
email	character varying(100)	"xxor_so677_run@gmail.com"
phone	character varying(20)	"87888965230"
photo_path	character varying(255)	"dwqfc725-2f06-2se4-9f6w-k836jjfk644n.jpg"

Атрибуты таблиц, отвечающие за консультации и записи, содержат тип данных, хранящее локальное время, могут понадобиться в разработке ПО.

Таблица 21 – Атрибуты таблицы consultation_templates.

Название атрибута	Тип данных	Пример данных
id	integer	105
teacher_id	integer	1
day_of_week	character varying(10)	"FRIDAY"
start_time	time without time zone	"13:36:00"
end_time	time without time zone	"13:37:00"
room	character varying(50)	"101"
max_students	integer	5
created_at	timestamp without time zone	"2025-05-23 13:34:47.170435"
updated_at	timestamp without time zone	"2025-05-23 13:34:47.170435"
is_one_time	boolean	false
expiration_datetime	timestamp without time zone	"2025-05-23 13:36:14.568136"
is_bi_weekly	boolean	true
start_next_week	boolean	false

Таблица 22 – Атрибуты таблицы consultations

Название атрибута	Тип данных	Пример данных
id	integer	217
template_id	integer	106
teacher_id	integer	1
consultation_date	date	"2025-05-30"
start_time	time without time zone	"13:37:00"
end_time	time without time zone	"13:38:00"
room	character varying(50)	"101"
max_students	integer	5
created_at	timestamp without time zone	"2025-05-23 13:35:21.269124"
updated_at	timestamp without time zone	"2025-05-23 13:35:21.269124"

Таблица 23 – Атрибуты таблицы consultation_signups

Название атрибута	Тип данных	Пример данных
id	integer	104
consultation_id	integer	217
student_id	integer	1
signup_time	timestamp without time zone	"2025-05-27 13:53:07.319351"
reason	text	"Сдать задолженность по предмету Автоматизация конструкторского проектирования ЭА"

Таблица 24 – Атрибуты таблицы consultation_archive

Название атрибута	Тип данных	Пример данных
id	integer	102
consultation_id	integer	217
student_id	integer	1
attended	boolean	true
feedback	text	"Сдал задолженность за 3 семестр по предмету АКП ЭА"
archived_at	timestamp without time zone	"2025-05-22 23:26:01.68853"
consultation_date	date	"2025-05-22"
start_time	time without time zone	"23:27:00"
end_time	time without time zone	"23:28:00"
room	character varying(50)	"101"
teacher_id	integer	1
reason	text	" Сдать задолженность по предмету Автоматизация конструкторского проектирования ЭА "
is_hidden	boolean	false
debt_status	character varying(25)	Сдал задолженность

Таким образом типы связей отражают отношения между сущностями, то есть как данные в одной таблице соотносятся с данными в другой.

2.4 Моделирование структуры поведения системы

Разрабатываемая система имеет несколько типов пользователей, каждый из которых выполняют свои задачи и функции, что естественно усложняет разработку. Без представления взаимодействия пользователей, структуры системы, описания бизнес-процессов, изменения состояний в зависимости от событий, из каких компонентов состоит программный продукт, на разработку ПО будет тратиться больше времени, а именно на исправление ошибок, так как отсутствует четкий план действий. Это может привести к противоречиям, а код станет менее поддерживаемым и расширяемым.

Для глубокого понимания взаимодействия компонентов системы применяется моделирование поведения и структуры системы посредством стандартизированного языка графического описания для объектного моделирования – UML [11]. У UML-диаграмм широкая классификация и для разрабатываемой системы будет использоваться как структурные, так и поведенческие диаграммы, которые подразделяются на несколько видов.

Правильным решением будет начать с того, как внешние сущности взаимодействуют с системой. На этапе проектирования было рассмотрено, какие таблицы и атрибуты существуют в системе, что создает некий каркас для того, чтобы показать, как эти таблицы используются в реальных сценариях. Поэтому для представления о функциональности системы будет полезна диаграмма вариантов использования (диаграмма прецедентов), относящаяся к поведенческим диаграммам, которая поможет избежать упущения и упростить предстоящую разработку и тестирование ПО.

Функции данного клиент-серверного приложения подразделяются на две группы: для неавторизованного и авторизованного пользователя. Но так как неавторизованный пользователь обладает меньшим количеством задач, то оптимальным выбором будет рассмотреть сценарии для авторизованного участника системы.

Поскольку в системе присутствуют несколько типов пользователей, то лучше будет разделить эту диаграмму на три части: для преподавателя, студента

и руководства.

У всех авторизованных пользователей есть следующие возможности:

- управление профилем, которое включает просмотр информации профиля, редактирование контактных данных, расширяющее изменение почты и телефона, и загрузку фотографии профиля;
- выход из системы, позволяющий перейти на страницу неавторизованного пользователя.

Перед авторизованным преподавателем стоит множество задач, представленных на рисунке 7.



Рисунок 7 – UML-диаграмма вариантов использования для преподавателя

Преподаватель выполняет следующие функции:

- управление консультациями, включающее создание регулярной консультации с настройкой (указания дня недели, консультации по 1 и (или) 2 неделе, времени начала, времени окончания, аудитории, максимального количества студентов, а также необязательной даты и времени удаления), создания разовой консультации с настройкой (указания даты, времени начала, времени окончания, аудитории, максимального количества студентов), просмотр списка консультаций, расширяющее удаление регулярного шаблона и конкретной консультации, включающее разовую консультацию;

- просмотр записи студентов, в котором будет содержаться ФИО студента, группа и причина записи;
- управление архивными записями студентов, включающее просмотр прошедших консультаций студентов, редактирования статуса посещения, сдачи задолженности и оставления отзыва, и расширяющее возможность скрытия записи.

Требования к системе, которые предъявляются студенту, проиллюстрированы на рисунке 8.



Рисунок 8 – UML-диаграмма вариантов использования для студента

Студент обладает следующими возможностями:

- запись на консультацию, которая подразумевает поиск преподавателя по ФИО и просмотру всех доступных консультаций, на которую можно записаться, указав причину записи;
- просмотр архивных записей, которые являются прошедшими;
- управление оповещениями, включающее отметку одного или нескольких уведомлений как прочитанных.

У руководства учебного подразделения (деканата) большинство функций, которые представлены на рисунке 9. Данный тип пользователя не предназначен для внесения каких-либо изменений в систему, в отличие от студентов и преподавателей. Руководство учебного подразделения ведет контроль над успеваемостью студентов.



Рисунок 9 – UML-диаграмма вариантов использования для руководства

Перед руководством стоят следующие задачи:

- просмотр информации о студентах, который представляет собой поиск студента, проверку архивных записей и просмотр статистики, включающий анализ посещаемости и сдачу задолженностей;
- просмотр информации о преподавателях, предполагающий поиск преподавателя, контроль архивных записей и просмотр статистики, расширяющий возможность проведения аналитики посещаемости студентов на консультации и сдачу их задолженностей.

После того, как был определен список актеров, которые будут использовать систему, и были выявлены, какие функции необходимы для удовлетворения потребностей пользователей, следующим этапом будет представление рабочих потоков и последовательность действий в системе, а именно логику выполнения бизнес-процессов. Это визуализирует диаграмма деятельности, которая позволяет оптимизировать процессы, так как помогает выявить ненужные шаги и избежать ошибки в сложных сценариях.

Разумным решением будет представить две UML-диаграммы деятельности: для создания консультации и запись на нее. Таким образом будет рассматриваться активность двух типов пользователей, а именно как со стороны преподавателя, так и со стороны студента, поскольку руководство системы не представляет функциональные изменения в системе.

На рисунке 10 представлена диаграмма деятельности, представляющая процесс создания консультации преподавателем.

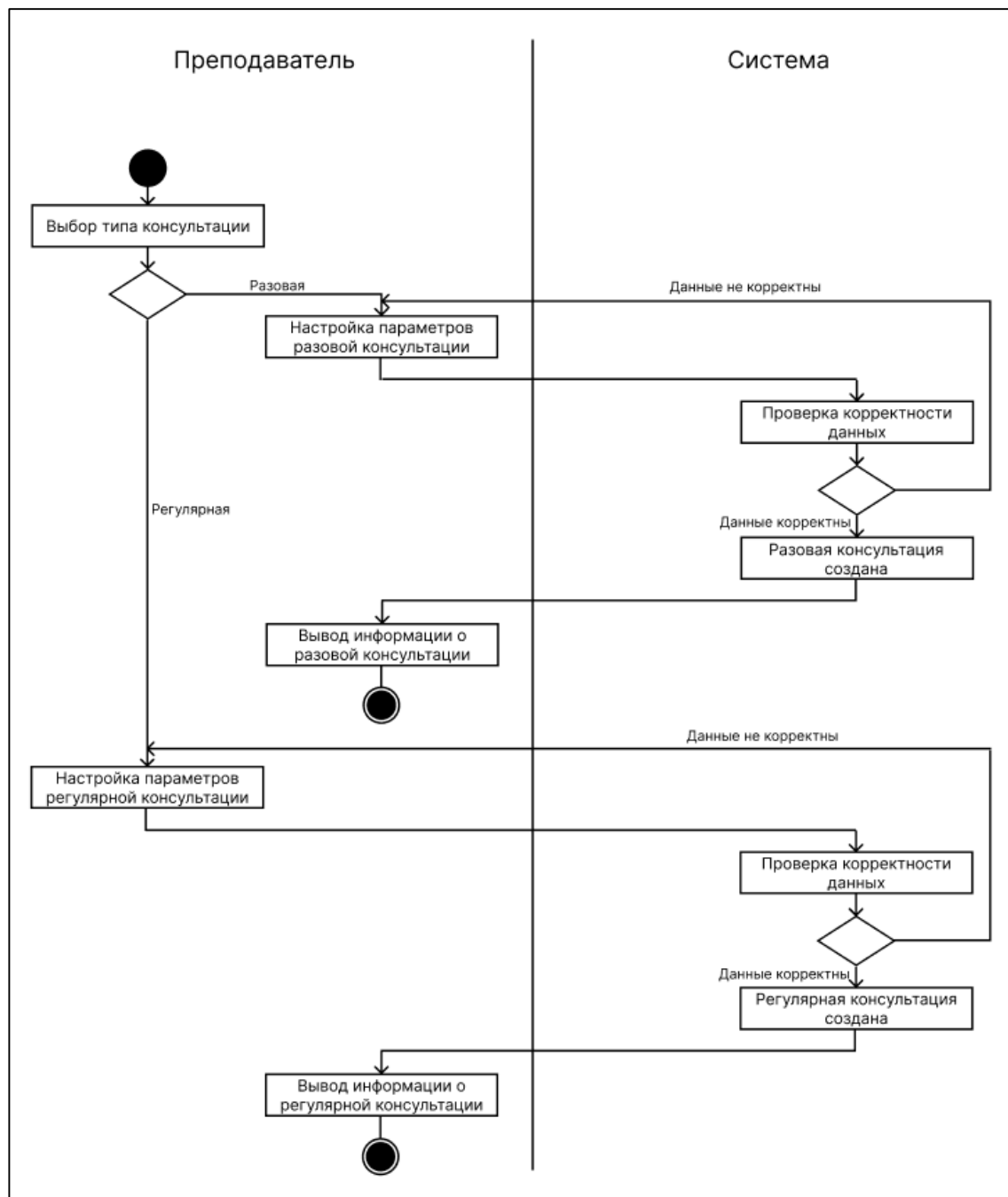


Рисунок 10 – UML-диаграмма деятельности преподавателя

У преподавателя стоит выбор между созданием двух типов консультаций: регулярной, предполагающая указание дня недели, по которым будут создаваться консультации, и разовой, где указывается конкретная дата. Настройками регулярной консультации, помимо указания дня, являются: создание консультации на каждую/через неделю, а также указание времени начала и окончания консультации, аудитории, количества человек, даты и времени удаления.

Параметры, которые необходимо будет указать при создании разовой консультации, включая дату, считаются: время начала и окончания конкретной консультации, аудитория и максимальное количество человек. Система должна предусмотреть проверку данных в формах ввода информации и проверить их валидность. Только в этом случае будет создана консультация.

Диаграмма деятельности, которая будет описывать процесс записи студента на консультацию, изображена на рисунке 11.

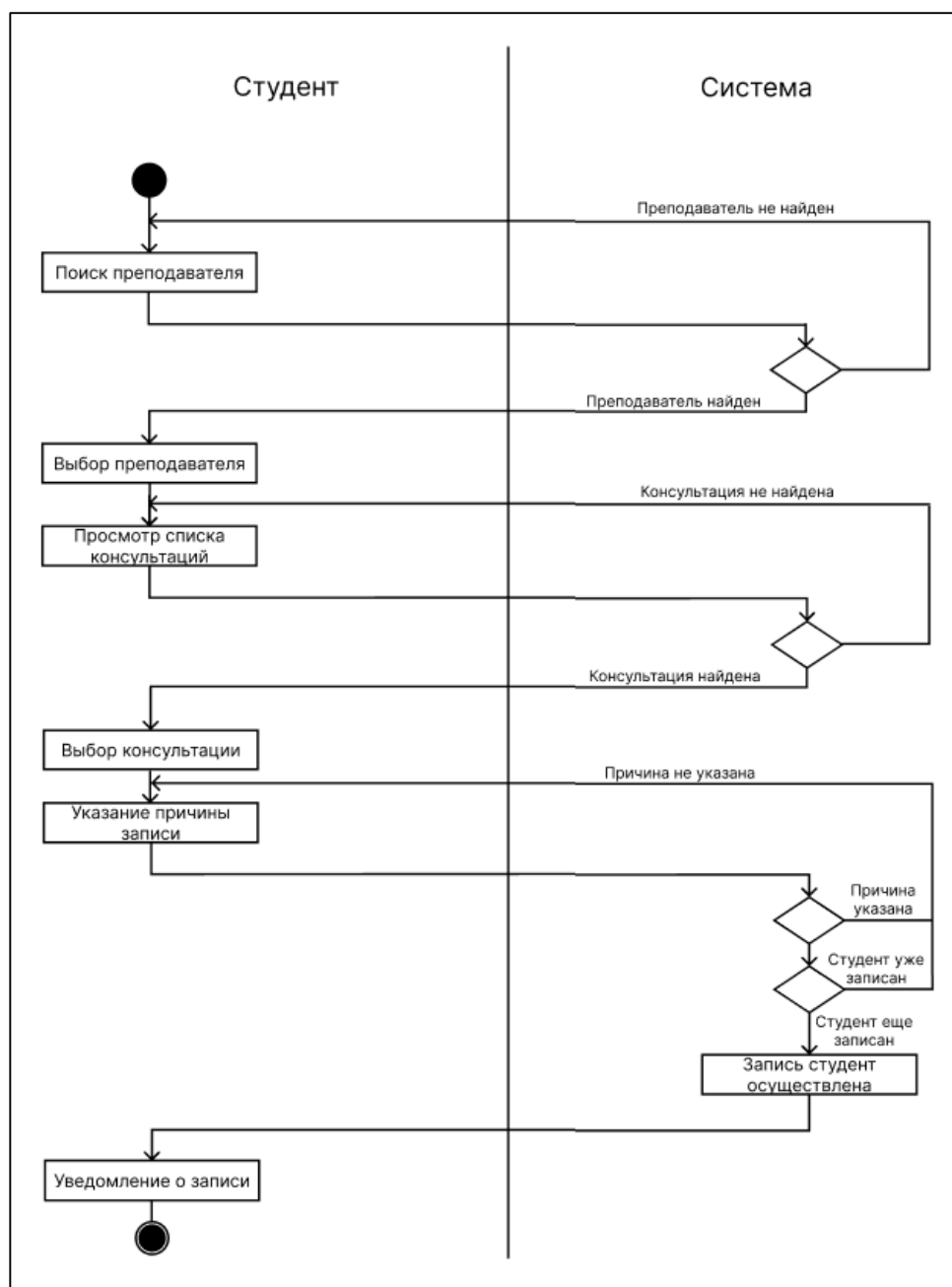


Рисунок 11 – UML-диаграмма деятельности студента

Чтобы записаться на консультацию, студенту нужно будет сделать поиск преподавателя и, если он найден, выбрать его и просмотреть список доступных

консультаций. Для записи необходимо будет указать причину посещения, что является обязательным условием, иначе процесс будет считаться не выполненным. Если студент еще не записан на эту консультацию, его запись подтверждается. В противном случае, запись не будет осуществлена, поскольку нельзя записаться на одну и ту же консультацию более одного раза.

В разделе, где рассматривались технологии и инструменты разработки, были выбраны сервер приложений Tomcat и СУБД PostgreSQL. А также есть клиент, о котором шла речь в архитектуре веб-приложения. Теперь нужно понять, как программные компоненты работают в реальной физической среде. Для этого нужно визуализировать физическую архитектуру системы на уровне оборудования и сетевых соединений. Для этого на рисунке 12 приведена диаграмма развертывания.

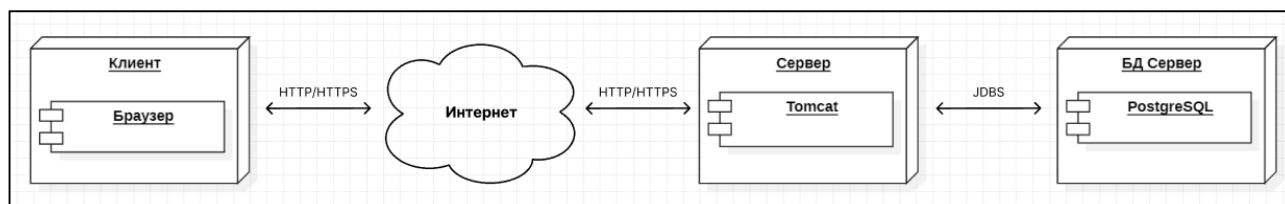


Рисунок 12 – UML-диаграмма развертывания системы

На диаграмме представлены узловые компоненты: клиент, сервер приложений, сервер БД, а также интернет. Важно понимать, как эти узлы связаны между собой.

Связь между браузером и интернетом осуществляется через протокол HTTP и HTTPs, если рассматривать их на прикладном уровне. Протоколы отвечают за передачу данных, позволяя пользователям запрашивать и получать веб-страницы, поэтому считаются основными протоколами взаимодействия.

При взаимодействии веб-сервера с интернетом так же используются протоколы HTTP или HTTPs для передачи данных.

Что касается связи между веб-сервером и сервером БД, то здесь применяются различные протоколы. Но так как приложение написано на языке Java, применяется JDBS, представляющий собой стандарт взаимодействия Java-приложений с РСУБД.

3 Разработка программного обеспечения

После этапа проектирования идет непосредственно разработка, являющаяся ключевым этапом жизненного цикла ПО. Она включает в себя несколько задач: создание БД, подготовку среды разработки, реализация структуры проекта, выбор архитектурного паттерна и написание кода, приведенное в Приложении.

3.1 Создание БД

На этапе проектирования были описаны таблицы и их атрибуты. Теперь необходимо перейти в создание уже готовой БД для системы посредством написания SQL-запросов.

Вот пример SQL-кода создания таблицы пользователей в БД:

```
CREATE TABLE users (  
    id SERIAL PRIMARY KEY,  
    login VARCHAR(50) UNIQUE NOT NULL,  
    password_hash VARCHAR(100) NOT NULL,  
    role VARCHAR(20) NOT NULL CHECK (role IN ('Студент',  
'Преподаватель', 'Руководство'))  
);
```

В данном запросе есть ограничение поля role, которое может быть одним из указанных значений: студентом, преподавателем и руководством.

Так как система не использует регистрацию пользователей, а содержит для них уже существующие данные, необходимо внести информацию в созданную таблицу. Это можно сделать с помощью SQL-запросов, либо посредством графического интерфейса.

Вот пример внесения данных в таблицу через SQL-запрос:

```
INSERT INTO users (login, password_hash, role)  
VALUES ('student1',  
'$2a$10$SpA0tVpN2nvedgXlVC8yWN.ZmSEcRU1AQMg9GmbwgBEbimV4cfIxOa', 'Студент');
```

Убедиться в том, что данные действительно внесены, можно с помощью графического интерфейса, изображенного на рисунке 13.

	id	login	password_hash	role
	[PK] integer	character varying (50)	character varying (255)	character varying (20)
1	1	student1	\$2a\$10\$SpA0tVpN2n...	Студент

Рисунок 13 – UML-диаграмма развертывания системы

Схема таблиц и атрибутов представлена на рисунке 14.

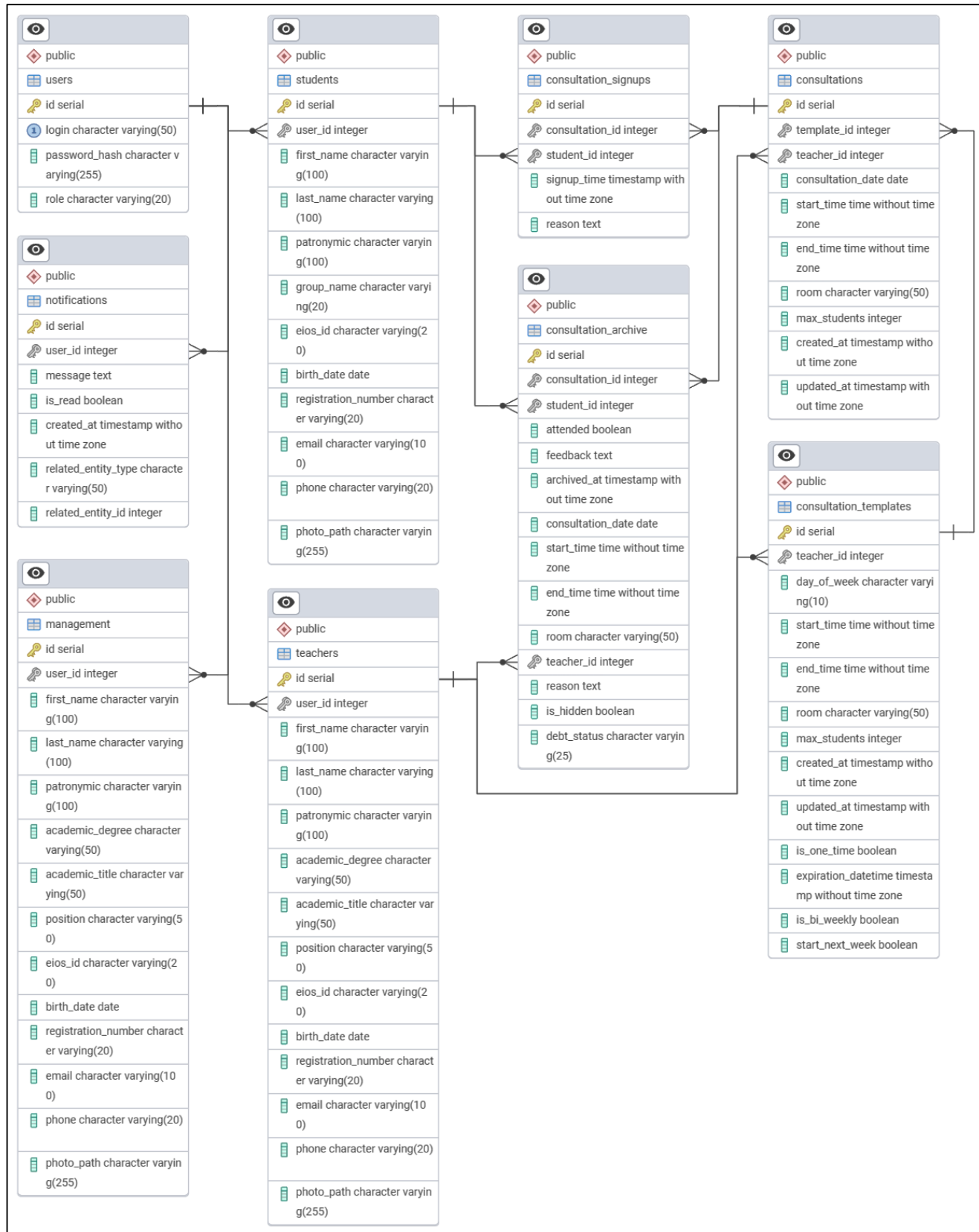


Рисунок 14 – Схема таблиц и атрибутов

На схеме уже явно продемонстрировано создание сущностей и полей, определены типы данных, индексы и ограничения целостности. На этом создание БД завершено.

3.2 Подготовка среды разработки

На этапе проектирования были выбраны технологии и инструменты разработки. И нужно убедиться, что они задействованы в наш проект. Речь идет об установке JDK и инструмента сборки проектов Maven в интегрированную среду разработки IntelliJ IDEA, поддерживающая фреймворк Spring Framework и его модули SpringBoot, Spring Security, Spring Data JPA, Spring MVC, которые будут настроены в файле pom.xml. Приведенная ниже зависимость указывает на использование Spring Boot Starter.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <version>3.3.4</version>
</dependency>
```

Помимо этой зависимости в проекте используются еще другие, которые связаны с использованием приведенных ранее модулей.

3.3 Реализация структуры проекта

На данном этапе необходимо организовать код так, чтобы он представлял логичное разделение проекта на компоненты, каждый из которых выполняет свою задачу. Таким образом можно добиться масштабируемой, поддерживаемой и проще тестируемой в дальнейшем программы. Проект будет разделен на следующие слои:

– модели, описывающие структуры данных, которые соответствуют количеству сущностей в БД, а поля точно соответствуют колонкам в таблице, и содержит аннотацию @Entity:

```
@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id; @Column(unique = true, nullable = false)
    private String login; @Column(nullable = false)
    private String passwordHash; @Column(nullable = false)
    private String role;
}
```


– репозитории, ориентированные на работу с базой данных, позволяющие выполнять сложные запросы, которые с использованием Spring Data JPA будут реализовываться в несколько строк кода:

```
public interface UserRepository extends JpaRepository<User, Integer> {  
    Optional<User> findByLogin(String login); // автоматизирует SELECT  
* FROM users WHERE login = ?  
}
```

– конфигурация, включающая настройку приложения, как например эту, которая используется для загрузки и отображения пользовательских фотографий, хранящиеся на сервере:

```
@Override  
public void addResourceHandlers(ResourceHandlerRegistry registry) {  
    registry.addResourceHandler("/uploads/**")  
        .addResourceLocations("file:uploads/");  
}
```

– контроллеры, имеющие аннотацию `@Controller`, которые принимают HTTP-запросы, проверяют корректность входных данных, вызывают сервисы для обработки запроса и формируют ответ;

– сервисы, содержащие аннотацию `@Service`, представляющие бизнес-логику приложения;

– файлы, отвечающие за пользовательский интерфейс (HTML, CSS, JS).

Примеры контроллеров, сервисов и других слоев будут представлены более подробно в Приложении.

На рисунке 15 схематично представлена структура проекта, которая содержит все основные компоненты, грамотно распределенные для минимизации потенциальных угроз.

3.4 Выбор архитектурного паттерна

В предыдущем подразделе уже можно догадаться, что для разработки ПО был выбран архитектурный паттерн MVC. Он очень хорошо организует код и разделяет разрабатываемое веб-приложение на три компонента, о которых так же было сказано ранее, каждый выполняющий свою определенную задачу: модель (управляет данными), представление (отвечает за отображение данных) и контроллер (обеспечивает взаимодействие пользователя с системой).

					ВКР 090301.06 25 81 01	Лист
Изм.	Лист	№ докум.	Подпись	Дата		49

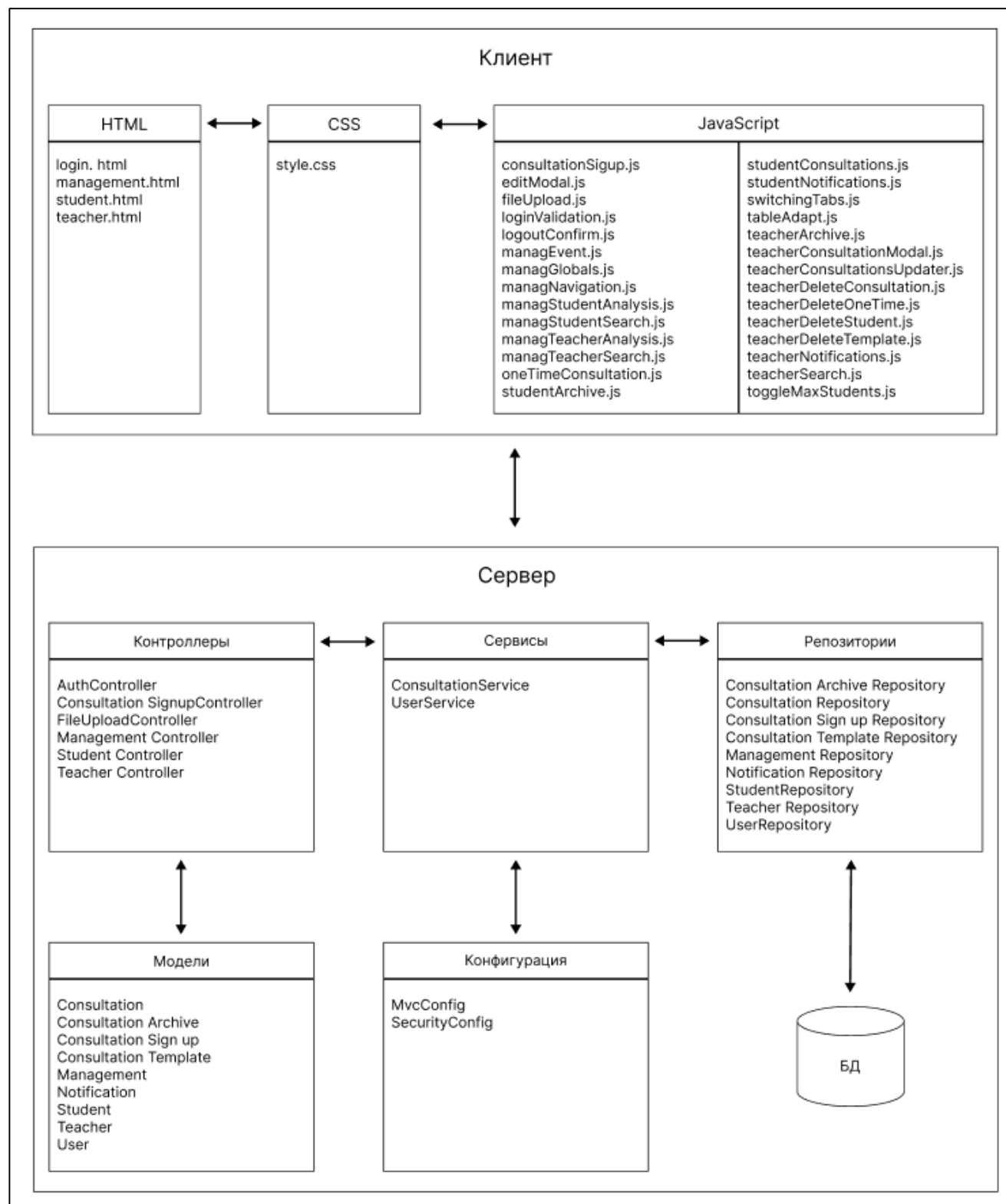


Рисунок 15 – Структура проекта

Помимо разделения частей проекта, данная структура четко отображает, как компоненты системы взаимодействуют друг с другом.

4 Тестирование программного обеспечения

Завершающим этапом разработки ПО является тестирование программного продукта. В этом разделе будут приведены несколько основных тестов:

- создание и удаление консультации преподавателем;
- создание и удаление записи студентом;
- просмотр статистики студентов и преподавателей руководством;
- адаптация под мобильные устройства.

Основной целью тестирования является обнаружение ошибок и их исправление для обеспечения высокого качества программного продукта.

4.1 Создание и удаление консультации преподавателем

Для начала проводится тестирование создания регулярной консультации преподавателем. Если какие-то данные в форме не будут указаны, система предупредит об этом пользователя. Как например ситуация, представленная на рисунке 16, которая требует ввода дня недели консультации.

Создание регулярной консультации

День недели: Выберите день

Выберите один из пунктов списка.

Время начала: --:--

Время окончания: --:--

Аудитория:

Макс. количество студентов: 5

☐ Без ограничения количества студентов

Дата автоматического удаления (необязательно): ДД.ММ.ГГГГ

Время автоматического удаления (необязательно): --:--

Создать консультацию

Рисунок 16 – Неполный ввод данных в форму

Как уже было сказано ранее, у преподавателя есть возможность удаления шаблона регулярной консультации в случае ввода даты и времени. На рисунке 17

система уведомляет пользователя о том, что при удалении консультации необходимо указать два параметра.

The screenshot shows a user profile on the left with a photo and contact information. The main area contains a consultation form. A modal notification is displayed in the center, stating: "Уведомление от сайта localhost. Для автоматического удаления необходимо указать и дату, и время." (Notification from the localhost site. For automatic deletion, you must specify both date and time.) The form fields include: "Время окончания:" (13:45), "Время окончания:" (15:20), "Аудитория:" (7а-203), "Макс. количество студентов:" (10), "Дата автоматического удаления (необязательно):" (30.06.2025), and "Время автоматического удаления (необязательно):" (---:--). A "Заккрыть" (Close) button is visible in the modal.

Рисунок 17 – Неполный ввод данных в форму

Если неправильно указан диапазон времени, система оповестит об этом преподавателю. Данная ситуация проиллюстрирована на рисунке 18.

The screenshot shows a modal notification stating: "Уведомление от сайта localhost. Время начала должно быть раньше времени окончания." (Notification from the localhost site. Start time must be earlier than end time.) The form fields show "Время начала:" (11:00) and "Время окончания:" (10:00). A "Заккрыть" (Close) button is visible in the modal.

Рисунок 18 – Некорректный ввод данных в форму

При корректности данных и нажатии на кнопку «Создать консультацию» на рисунке 19 преподаватель получает уведомление об успешном создании шаблона.

The screenshot shows a modal notification stating: "Уведомление от сайта localhost. Шаблон регулярной консультации успешно создан." (Notification from the localhost site. Regular consultation template successfully created.) A "Заккрыть" (Close) button is visible in the modal.

Рисунок 19 – Уведомление об успешном создании регулярного шаблона

По неделе будет определена ближайшая дата конкретной консультации.

Если консультация прошла, то автоматически будет генерироваться следующая. Если на конкретную консультацию записалось максимальное количество человек, то студентам предоставляется следующая дата для записи.

На рисунке 20 идет создание уже другого типа консультации – разовой, где преподаватель указывает дату проведения на один раз, но в приведенной ситуации нужно учесть момент, что дата создания консультации может существовать в будущем и должна предшествовать.

Создание разовой консультации

Дата консультации: 02.05.2025

Минимальное значение — 28.05.2025.

09:50

Время окончания: 11:25

Аудитория: 7a-202

Макс. количество студентов: 10

☒ Без ограничения количества студентов

Создать консультацию

Регулярные консультации

ДЕНЬ НЕДЕЛИ	ВРЕМЯ НАЧАЛА
Понедельник	13:45

Консультация: Понедельник 13:45- 15:00

ДАТА	ВРЕМЯ НАЧАЛА
02.06.2025	13:45

Разовые консультации

Уведомления 1

ДАТА ЗАВЕРШЕНИЯ	ДЕЙСТВИЕ
30.06.2025 10:00	Удалить

КС. СТУДЕНТОВ

ДЕЙСТВИЕ
Удалить

Разовых консультаций пока нет

Рисунок 20 – Некорректный ввод данных в форму

В случае валидности данных будет создано оповещение, продемонстрированное на рисунке 21.

Уведомление от сайта localhost

Разовая консультация успешно создана

Заккрыть

Регулярные консультации

Уведомления 1

Рисунок 21 – Уведомление об успешном создании разовой консультации

В базе данных добавлена информация о шаблоне регулярной консультации,

представленная на рисунке 22.

Data Output Messages Notifications									
Showing rows: 1 to 1 Page No: 1 of 1									
	id [PK] integer	teacher_id integer	day_of_week character varying (10)	start_time time without time zone	end_time time without time zone	room character varying (50)	max_students integer	created_at timestamp without time zone	
1	109	1	MONDAY	13:45:00	15:20:00	7a-203	10	2025-05-28 03:45:01.213615	

Рисунок 22 – Внесение данных в БД

А также есть добавление информации о конкретных консультациях, приведенная на рисунке 23.

Data Output Messages Notifications									
Showing rows: 1 to 2 Page No: 1 of 1									
	id [PK] integer	template_id integer	teacher_id integer	consultation_date date	start_time time without time zone	end_time time without time zone	room character varying (50)	max_students integer	
1	221	109	1	2025-06-02	13:45:00	15:20:00	7a-203	10	
2	222	[null]	1	2025-06-30	09:50:00	11:25:00	7a-202	10000	

Рисунок 23 – Внесение данных в БД

На странице преподавателя будет отображаться вся необходимая информация о консультации. Это можно увидеть на рисунке 24.

Консультация

Записи

Архив записей

Уведомления1

Мои шаблоны консультаций

Создать регулярную консультацию

Создать разовую консультацию

Регулярные консультации

ДЕНЬ НЕДЕЛИ	ВРЕМЯ НАЧАЛА	ВРЕМЯ ОКОНЧАНИЯ	АУДИТОРИЯ	МАКС. СТУДЕНТОВ	ДАТА ЗАВЕРШЕНИЯ	ДЕЙСТВИЕ
Понедельник	13:45	15:20	7а-203	10	30.06.2025 10:00	Удалить

Консультация: Понедельник 13:45- 15:20

ДАТА	ВРЕМЯ НАЧАЛА	ВРЕМЯ ОКОНЧАНИЯ	АУДИТОРИЯ	МАКС. СТУДЕНТОВ	ДЕЙСТВИЕ
02.06.2025	13:45	15:20	7а-203	10	Удалить

Разовые консультации

ДАТА	ВРЕМЯ НАЧАЛА	ВРЕМЯ ОКОНЧАНИЯ	АУДИТОРИЯ	МАКС. СТУДЕНТОВ	ДЕЙСТВИЕ
30.06.2025	09:50	11:25	7а-202	не ограничено	Удалить

Рисунок 24 – Отображение данных о созданных консультациях

На рисунке 25 есть отображение уведомления о дате и времени удаления регулярного шаблона.

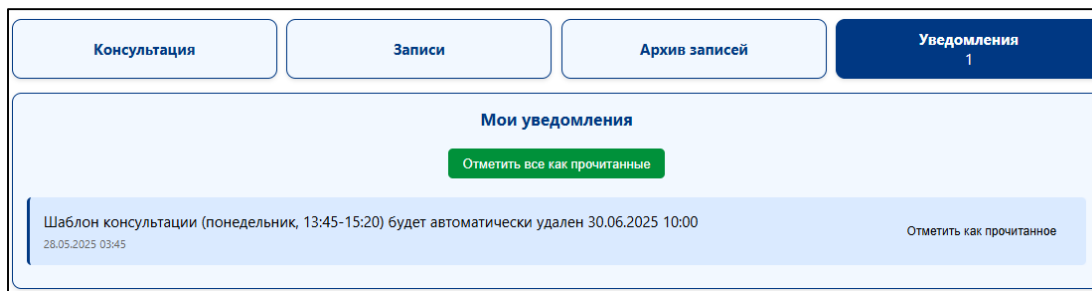


Рисунок 25 – Отображение уведомления об изменениях

Затем рассматривается возможность удаления регулярного шаблона, при котором все конкретные консультации по этому шаблону будут удалены.

Уведомление об удалении конкретной консультации по регулярному шаблону приведена на рисунке 26.

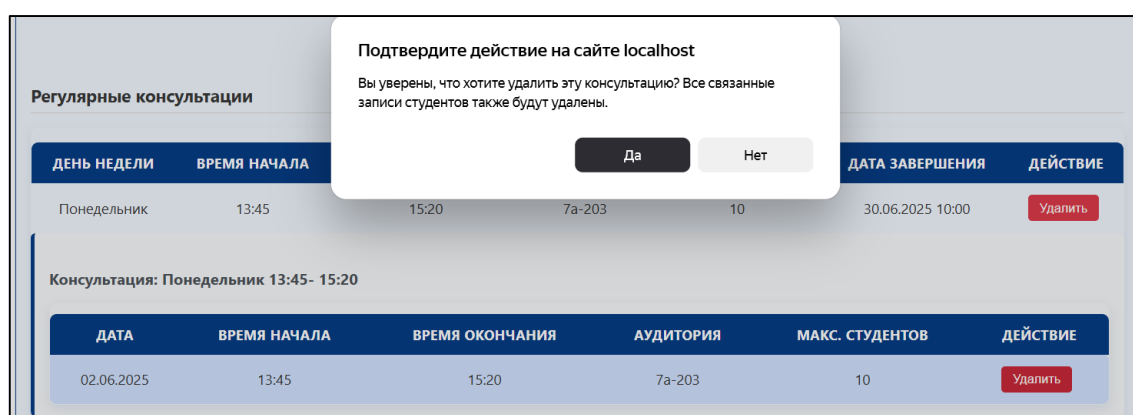


Рисунок 26 – Отображение уведомления об удалении

Генерация следующей даты указанной недели представлена на рисунке 27.

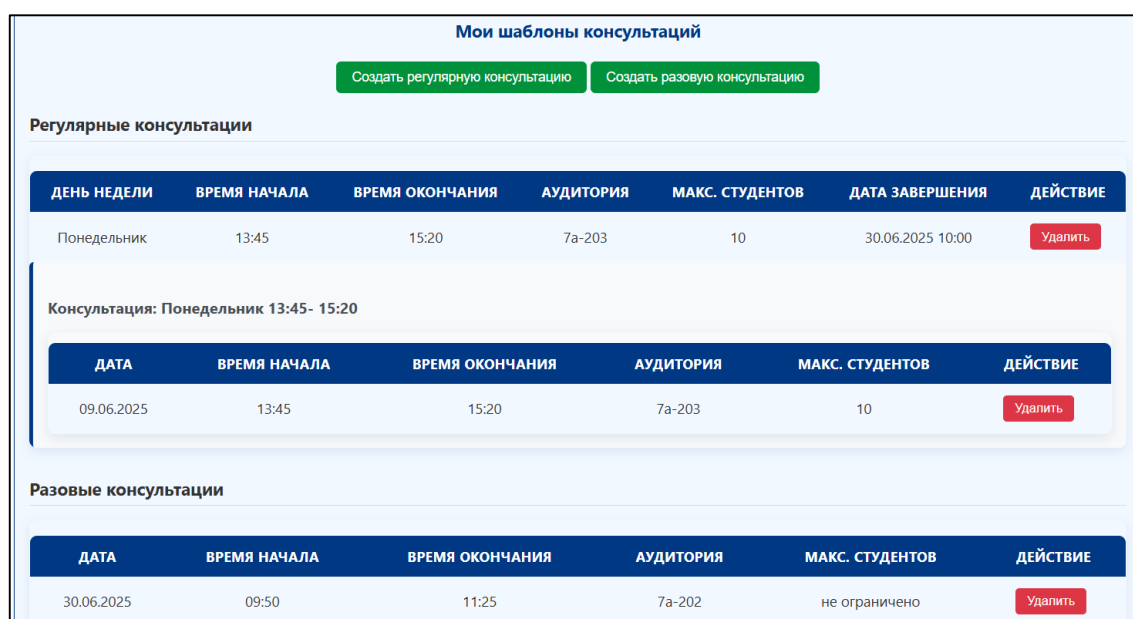


Рисунок 27 – Отображение данных о созданных консультациях

На рисунке 28 проиллюстрировано уведомление о подтверждении удаления

шаблона регулярной консультации.

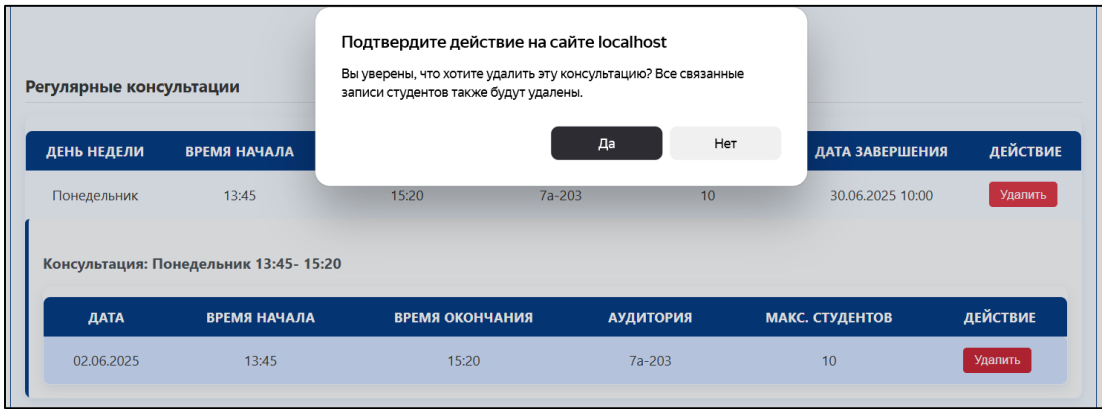


Рисунок 28 – Отображение уведомления об удалении

После удаления на рисунке 29 можно заметить, что регулярная консультация отсутствует.

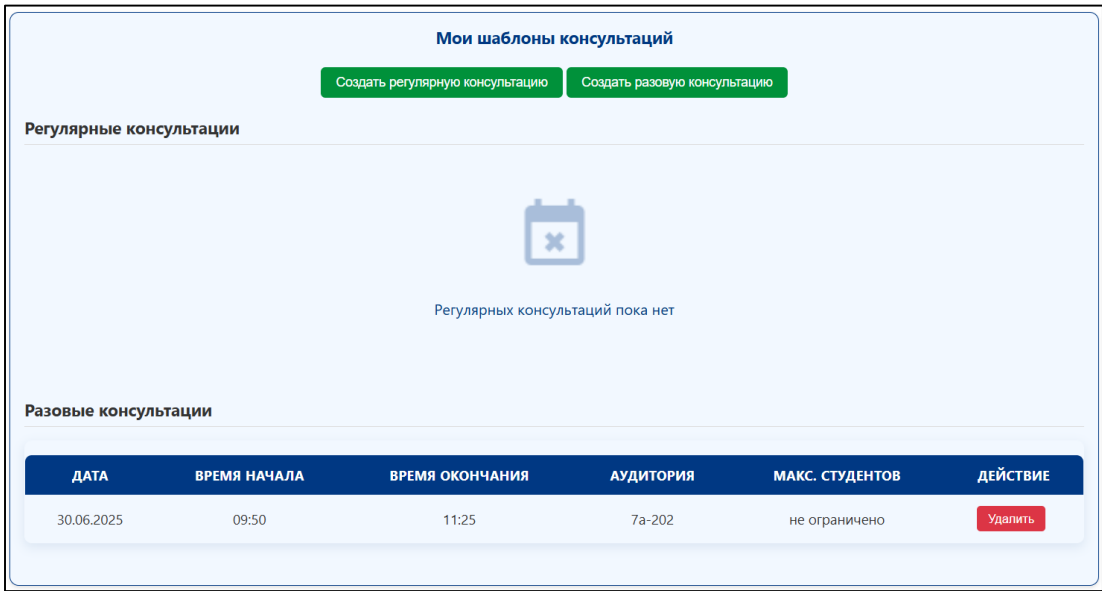


Рисунок 29 – Отображение данных о созданных консультациях

Уведомление о подтверждении удаления разовой консультации отображено на рисунке 30.

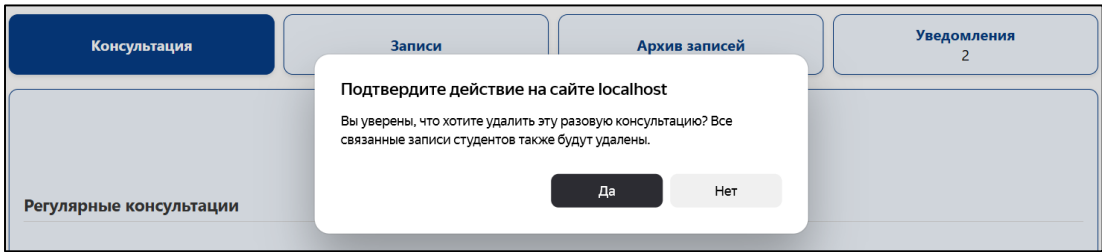


Рисунок 30 – Отображение данных о созданных консультациях


На рисунке 31 приведено уведомление о подтверждении удаления шаблона регулярной консультации.

Мои шаблоны консультаций

Создать регулярную консультацию


Создать разовую консультацию

Регулярные консультации



Регулярных консультаций пока нет

Разовые консультации



Разовых консультаций пока нет

Рисунок 31 – Отсутствие данных о консультациях


На этом тестирование процесса создания и удаления консультаций преподавателем завершено.

4.2 Создание и удаление записи студентом

На рисунке 32 представлено, что на данный момент студент пока что не записан ни на одну консультацию.

Личный кабинет студента

Профиль



Загрузить фото

ID ЭИОС: 517710386

Контакты

Почта: nda03@yandex.ru

Изменить

Телефон: 86562651515

Изменить


Консультация

Записи

Уведомления

Мои записи на консультации

Записаться на консультацию



У вас нет записей на консультации

Подобная информация

Фамилия: Нагорная

Имя: Дарья

Отчество: Александровна

Дата рождения: 19.06.2003

Табельный/регистрационный номер: 72482

Группа: 21BBC1

Рисунок 32 – Отсутствие данных о записях студента

Для того, чтобы записаться на консультацию, студенту необходимо найти

преподавателя через поиск, в котором нужно ввести его ФИО. На рисунке 33 показана ситуация, когда с БД нет информации о данном пользователе.

The screenshot shows a web interface with three tabs: 'Консультация' (active), 'Записи' (Records), and 'Уведомления' (Notifications). Below the tabs is a breadcrumb trail: 'Профиль >> Поиск преподавателя'. A search input field contains the text 'Сергей'. Below the input field, the message 'Преподаватель не найден' is displayed.

Рисунок 33 – Отсутствие данных о преподавателе

При вводе существующих данных о преподавателе, которые присутствуют в БД, он будет отображен, как это видно на рисунке 34.

The screenshot shows the same web interface as Figure 33. The search input field now contains 'Кузнец'. Below the input field, a search result is displayed: 'Кузнецов Дмитрий Владимирович' followed by 'Заведующий кафедрой, Профессор'.

Рисунок 34 – Отсутствие данных о преподавателе

На рисунке 35 представлены все доступные консультации преподавателя, на которые может записаться студент.

The screenshot shows the 'Консультация' page with a breadcrumb trail: 'Профиль >> Поиск преподавателя >> Консультации преподавателя'. The teacher's name 'Кузнецов Дмитрий Владимирович' is displayed. Below the name, two consultation slots are listed:

- Slot 1:
 - Дата: 02.06.2025
 - Время: 13:45:00 - 15:20:00
 - Аудитория: 7а-203
 - Записано: 0/10
 - Button: 'Записаться на консультацию' (green)
- Slot 2:
 - Дата: 30.06.2025
 - Время: 09:50:00 - 11:25:00
 - Аудитория: 7а-202
 - Записано: 0 (не ограничено)
 - Button: 'Записаться на консультацию' (green)

Рисунок 35 – Отображение данных о доступных консультациях

Для записи студента необходимо указать причину записи. Если это не будет сделано, то система будет предупреждать об этом студента, как это отображено на

рисунке 36.

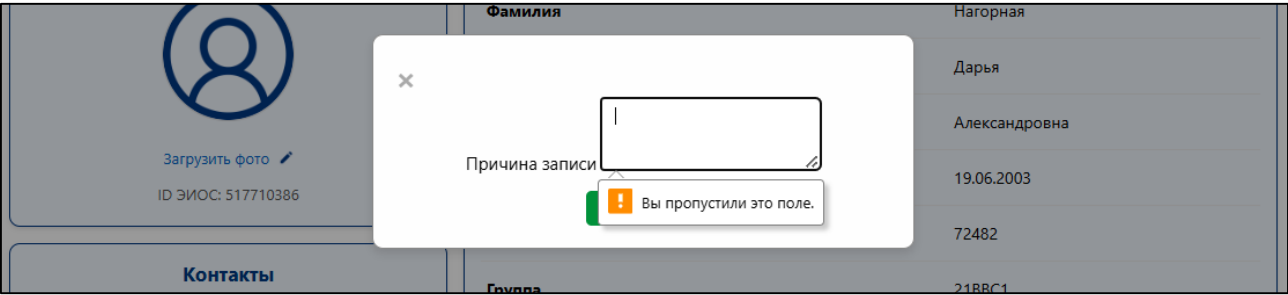


Рисунок 36 – Отсутствие данных в поле

На рисунке 37 приведен пример заполнения причины записи.

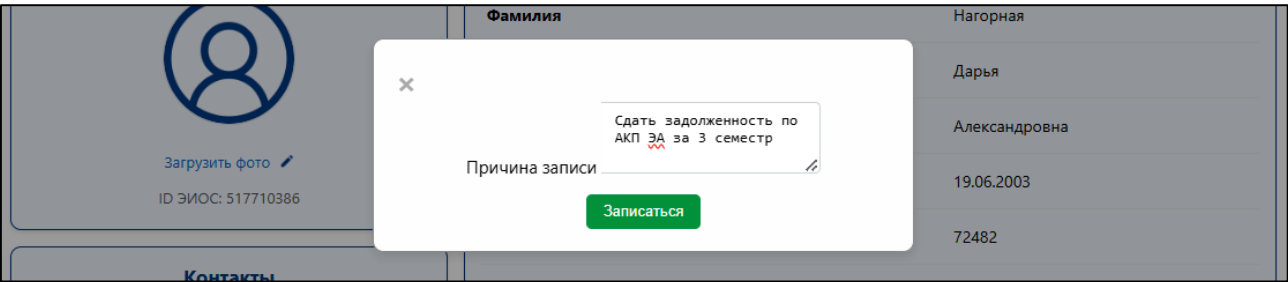


Рисунок 37 – Ввод данных в поле

Уведомление об успешной записи приведено на рисунке 38.

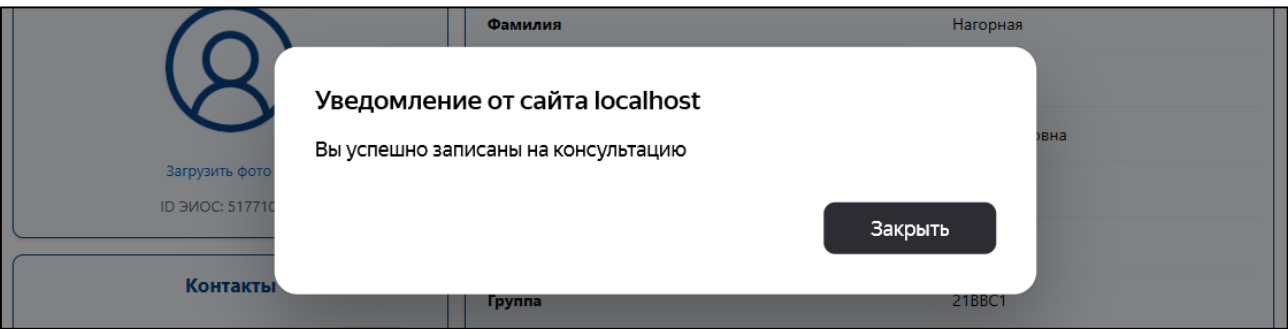


Рисунок 38 – Уведомление на странице студента

В случае повторной записи система будет оповещать студента о невыполнении данной задачи, как это проиллюстрировано на рисунке 39.

Data Output Messages Notifications						
Showing rows: 1 to 1						
	id [PK] integer	consultation_id integer	student_id integer	signup_time timestamp without time zone	reason text	
1	106	221	1	2025-05-28 03:51:12.662022	Сдать задолженность по АКП ЭА за 3 семестр	

Рисунок 39 – Создание записи в БД

Отображение записавшегося студента видна на странице преподавателя и приведена на рисунке 40.

Консультация

Записи

Архив записей

Уведомления
1

Записи студентов

Дата: 02.06.2025
Время: 13:45 - 15:20
Аудитория: 7а-203
Макс. студентов: 10

ФАМИЛИЯ	ИМЯ	ОТЧЕСТВО	ГРУППА	ПРИЧИНА ЗАПИСИ	ДЕЙСТВИЕ
Нагорная	Дарья	Александровна	21ВВС1	Сдать задолженность по АКП ЭА за 3 семестр	Удалить

Дата: 30.06.2025
Время: 09:50 - 11:25
Аудитория: 7а-202
Макс. студентов: не ограничено

Нет записей на эту консультацию

Рисунок 40 – Отображение записей студентов на странице преподавателя

Затем рассмотрим ситуацию об удалении записи. Уведомление о подтверждении проиллюстрировано на рисунке 41.

Профиль

Подтвердите действие на сайте localhost

Вы уверены, что хотите отменить запись на эту консультацию?

Да Нет

Отчество

Александровна

Рисунок 41 – Уведомление на странице студента

На рисунке 42 видно, что запись студента была удалена.

Консультация

Записи

Уведомления

Мои записи на консультации

Записаться на консультацию

У вас нет записей на консультации

Рисунок 42 – Отсутствие данных о записях студента

Запись студента также удалена из БД, как это приведено на рисунке 43.

Data Output	Messages	Notifications
id [PK] integer	consultation_id integer	student_id integer
signup_time timestamp without time zone	reason text	

Рисунок 43 – Удаление записи в БД

Отображение записавшегося студента видно на странице преподавателя и приведено на рисунке 44.

Консультация

Записи

Архив записей

Уведомления 1

Записи студентов

Дата: 02.06.2025

Время: 13:45 - 15:20

Аудитория: 7а-203

Макс. студентов: 10

ФАМИЛИЯ	ИМЯ	ОТЧЕСТВО	ГРУППА	ПРИЧИНА ЗАПИСИ	ДЕЙСТВИЕ
Нагорная	Дарья	Александровна	21BBC1	Сдать задолженность по АКП ЭА за 3 семестр	Удалить

Дата: 30.06.2025

Время: 09:50 - 11:25

Аудитория: 7а-202

Макс. студентов: не ограничено

Нет записей на эту консультацию

Рисунок 44 – Отображение записей студентов на странице преподавателя

Об изменении статуса записи студента на странице преподавателя есть оповещение, проиллюстрированное на рисунке 45.

Консультация

Записи

Архив записей

Уведомления 2

Мои уведомления

Отметить все как прочитанные

Студент Нагорная Дарья Александровна отменил запись на вашу консультацию 2025-06-02 в 13:45

28.05.2025 03:55

Отметить как прочитанное

Шаблон консультации (понедельник, 13:45-15:20) будет автоматически удален 30.06.2025 10:00

28.05.2025 03:45

Отметить как прочитанное

Рисунок 45 – Уведомление на странице преподавателя

Таким образом, был протестирован процесс создания и удаления записи студента.

4.3 Просмотр статистики студентов и преподавателей руководством

На странице руководства надо проверить, как отображается статистика посещаемости и сдачи задолженностей на страницах студента и преподавателя.

На рисунке 46 приведена аналитика студента.

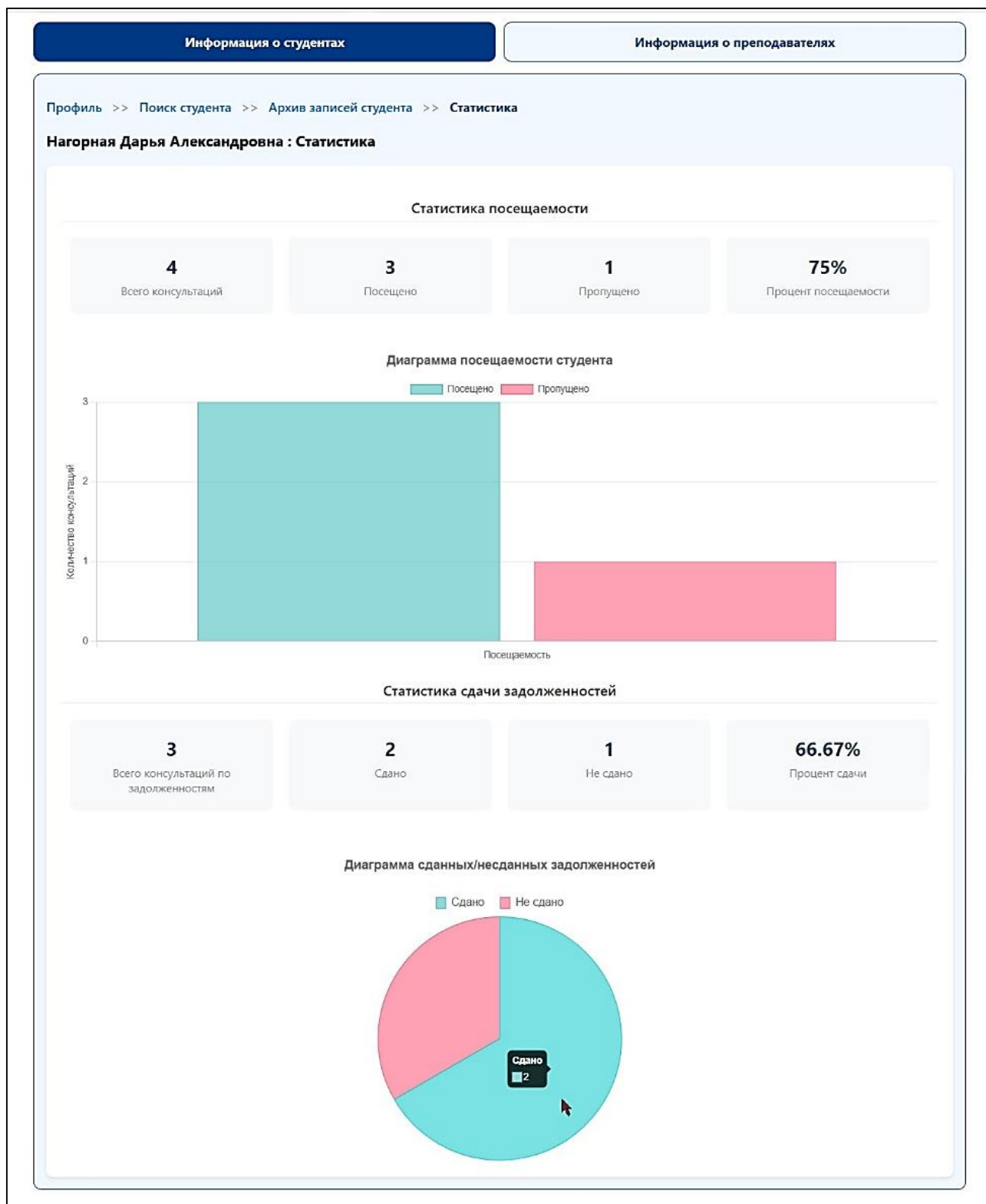


Рисунок 46 – Статистика студента

Аналитика преподавателя отображена на рисунке 47.



Статистика сдачи задолженностей

5

Всего консультаций по задолженностям

3

Сдано студентами

2

Не сдано студентами

60%

Процент сдачи

Диаграмма сданных/несданных задолженностей

Сдано

Не сдано

Статус	Количество задолженностей
Сдано	3
Не сдано	2

Рисунок 47 – Статистика преподавателя

В ходе тестирования просмотра статистики студентов и преподавателей на странице руководства было выявлено, что данные визуализированы корректно, то есть аналитика показывает правильную информацию.

4.4 Адаптация под мобильные устройства

Важным этапом создания пользовательского интерфейса является адаптация

под мобильные устройства. Поскольку данное ПО позволяет выполнять быстрые задачи, которые не требуют использования компьютера, более удобно использовать его на мобильных устройствах. Это в основном влияет на оптимизацию производительности веб-приложения и гибкую верстку.

На рисунке 48 приведена страница преподавателя. Здесь видно изменение структуры контента, а именно скрывание ненужной информации, изменение расположения элементов.

Регулярные консультации

День недели	Понедельник
Время начала	13:45
Время окончания	15:20
Аудитория	7а-203
Макс. студентов	10
Дата завершения	30.06.2025 10:00
Действие	Удалить

Консультация: Понедельник 13:45-15:20

Дата	02.06.2025
Время начала	13:45
Время окончания	15:20
Аудитория	7а-203
Макс. студентов	10
Действие	Удалить

Рисунок 48 – Адаптация под мобильные устройства

В ходе тестирования адаптации веб-приложения под мобильные устройства были изучен инструмент, позволяющий интегрировать внешний вид страниц под технические параметры устройства пользователя, который называется медиазапросом.

Заключение

В процессе выполнения выпускной квалификационной работы был проведен анализ предметной области и технического задания, а также выполнены проектирование, разработка и тестирование ПО.

В результате выполнения ВКР была разработана система онлайн-записи студентов на консультации к преподавателям, реализованная в виде веб-приложения.

Система позволяет автоматизировать организацию консультаций и составление расписания для преподавателей, запись студентов на консультации и управление ими, а также формирование отчета для руководства об академической активности студентов и аналитики работы преподавателей для выявления затруднений в учебном процессе и принятия результативных решений. Система не требует регистрации, так как представляется в виде подсистемы ЭИОС вуза и использует существующие учетные записи пользователей. У преподавателей, студентов и руководства есть возможность редактировать свои личные данные: фотографию профиля, почту и телефон.

Следовательно, можно сделать вывод, что разработанное веб-приложение улучшит взаимодействие между преподавателями, студентами и руководством и повысит качество учебного процесса, решая все проблемы, которые были обнаружены при обзоре аналогов.

Таким образом, требования технического задания выполнены в полном объеме.

					ВКР 090301.06 25 81 01	Лист
						65
Изм.	Лист	№ докум.	Подпись	Дата		

Список используемых источников

1. Коул А. Изучаем Flex 3. Руководство по разработке насыщенных интернет-приложений / А. Коул. – Москва : Символ-Плюс, 2017. – 795 с.
2. Нагорная Д. А., Гудков А. А. Система онлайн-записи студентов на консультации к преподавателям // Вестник Пензенского государственного университета. 2025. № 2. С. 79–81.
3. Московский государственный университет технологий и управления им. К. Г. Разумовского : официальный сайт. – URL: <https://mgutm.ru/> (дата обращения: 20.04.2025).
4. Южный федеральный университет : официальный сайт. – URL: <https://sfedu.ru/> (дата обращения: 29.04.2025).
5. Пензенский государственный университет : официальный сайт. – URL: <https://pnzgu.ru/> (дата обращения: 30.04.2025).
6. Лучшие языки для бэкенда: что выбрать в 2025 году [Электронный ресурс]. – URL: <https://ycla-coding.com/blog/luchshie-yazyki-dlya-bekenda-chto-vybrat-v-2025-godu/> (дата обращения: 01.05.2025).
7. Документация к фреймворку Spring Framework [Электронный ресурс]. – URL: <https://docs.spring.io/spring-framework/docs/4.2.x/spring-framework-reference/html> (дата обращения: 01.05.2025).
8. Курняван Б. Программирование Web-приложений на языке Java / Б. Курняван. – Москва : Лори, 2014. – 880 с.
9. Стоунз Р. PostgreSQL. Основы / Р. Стоунз, М. Нейл. – Санкт-Петербург : Символ-Плюс, 2013. – 640 с.
10. Клиент-серверная архитектура [Электронный ресурс]. – URL: https://ru.hexlet.io/courses/internet-fundamentals/lessons/client-server/theory_unit (дата обращения: 02.05.2025).
11. Буч Г. Язык UML. Руководство пользователя / Г. Буч, Дж. Рамбо, А. Джекобсон. – Москва : ДМК Пресс, 2019. – 432 с.

Приложение

(Обязательное)

Система онлайн-записи студентов на консультации к преподавателям

Листинг программы

					ВКР 090301.06 25 81 01	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

Файл ConsultationTemplate.java

```
package com.university.consultations.entity;

import jakarta.persistence.*;
import lombok.Data;
import java.time.DayOfWeek;
import java.time.LocalDateTime;
import java.time.LocalTime;

@Entity
@Table(name = "consultation_templates") // Таблица шаблонов
регулярной консультации
@Data
public class ConsultationTemplate {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @ManyToOne
    @JoinColumn(name = "teacher_id", nullable = false)
    private Teacher teacher; // Преподаватель

    @Column(nullable = false)
    @Enumerated(EnumType.STRING)
    private DayOfWeek dayOfWeek; // День недели

    @Column(nullable = false)
    private LocalTime startTime; // Время начала консультации

    @Column(nullable = false)
    private LocalTime endTime; // Время окончания консультации

    @Column(nullable = false)
    private String room; // Аудитория

    @Column(nullable = false)
    private Integer maxStudents; // Количество студентов

    @Column(nullable = false)
    private Boolean isOneTime = false;

    @Column(name = "expiration_datetime")
    private LocalDateTime expirationDateTime;

    @Column(name = "is_bi_weekly", nullable = false)
    private Boolean isBiWeekly = false;

    @Column(name = "start_next_week", nullable = false)
    private Boolean startNextWeek = false;
}
```

Файл Consultations.java

```
package com.university.consultations.entity;

import jakarta.persistence.*;
import lombok.Data;
import java.time.LocalDate;
```

```

import java.time.LocalDateTime;

@Entity
@Table(name = "consultations") // Таблица конкретных консультаций
@Data
public class Consultation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    @ManyToOne
    @JoinColumn(name = "template_id")
    private ConsultationTemplate template;

    @ManyToOne
    @JoinColumn(name = "teacher_id", nullable = false)
    private Teacher teacher; // Преподаватель

    @Column(nullable = false)
    private LocalDate consultationDate; // Дата консультации

    @Column(nullable = false)
    private LocalDateTime startTime; // Время начала консультации

    @Column(nullable = false)
    private LocalDateTime endTime; // Время окончания консультации

    @Column(nullable = false)
    private String room; // Аудитория

    @Column(nullable = false)
    private Integer maxStudents; // Количество студентов
}

```

Файл ConsultationSignups.java

```

package com.university.consultations.entity;

import jakarta.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table(name = "consultation_signups") // Таблица записей студентов
public class ConsultationSignup {

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

    @ManyToOne
    @JoinColumn(name = "consultation_id", nullable = false)
    private Consultation consultation; // Конкретная консультация

    @ManyToOne
    @JoinColumn(name = "student_id", nullable = false)
    private Student student; // Студент

    @Column(name = "signup_time", nullable = false)
    private LocalDateTime signupTime; // Дата и время создания записи
}

```

```

        @Column(name = "reason", nullable = false)
        private String reason; // Причинная записи
    }

```

Файл ConsultationService.java

```

package com.university.consultations.service;
import com.university.consultations.entity.*;
import com.university.consultations.repository.*;

import jakarta.transaction.Transactional;
import org.springframework.stereotype.Service;

import java.time.*;
import java.time.format.TextStyle;
import java.util.List;
import java.util.Locale;

@Service
public class ConsultationService { // Сервис для автоматической
генерации консультаций
    // Репозитории для работы с сущностями
    private final ConsultationTemplateRepository templateRepository;
    private final ConsultationRepository consultationRepository;
    private final ConsultationSignupRepository
consultationSignupRepository;
    private final ConsultationArchiveRepository
consultationArchiveRepository;
    private final NotificationRepository notificationRepository;
    public ConsultationService(ConsultationTemplateRepository
consultationTemplateRepository,
                                ConsultationRepository
onsultationRepository,
                                ConsultationSignupRepository
consultationSignupRepository,
                                ConsultationArchiveRepository
consultationArchiveRepository,
                                NotificationRepository
notificationRepository) {
        this.templateRepository = templateRepository;
        this.consultationRepository = consultationRepository;
        this.consultationSignupRepository =
consultationSignupRepository;
        this.consultationArchiveRepository =
consultationArchiveRepository;
        this.notificationRepository = notificationRepository;
    }
    // Вычисление следующей доступной даты для консультации по шаблону
    private LocalDate getNextAvDate(ConsultationTemplate template,
LocalDateTime now) {
        LocalDate today = LocalDate.now();
        DayOfWeek templateDay = template.getDayOfWeek();
        LocalDate nextDate = template.getStartNextWeek() ?
today.plusWeeks(1) : today;
        // Нахождение ближайшей даты недели, соответствующей шаблону
        while (nextDate.getDayOfWeek() != templateDay) {
            nextDate = nextDate.plusDays(1);
        }
    }

```

```

        // Перенос на следующую неделю/две недели консультации
        if (nextDate.isEqual(today) &&
template.getEndTime().isBefore(now.toLocalTime())) {
            nextDate = nextDate.plusWeeks(template.getIsBiWeekly() ?
2 : 1);
        } return nextDate;
    }
    //Удаление шаблона и отправка уведомления
    private void deleteTemplateNotifications(ConsultationTemplate
template) {
        // Уведомление для студентов
        List<Consultation> consultations =
consultationRepository.findByTemplateId(template.getId());
        consultations.forEach(consultation -> {
            List<ConsultationSignup> signups =
consultationSignupRepository.findByConsultationId(consultation.getId());
            signups.forEach(signup -> {
                Notification notification = new Notification();
                notification.setUser(signup.getStudent().getUser());
                notification.setMessage(String.format(
                    "Шаблон консультации с преподавателем %s %s
%s был автоматически удален по истечении срока действия",
                    template.getTeacher().getLastName(),
                    template.getTeacher().getFirstName(),
                    template.getTeacher().getPatronymic()
                ));
                notificationRepository.save(notification);
            });
        });
        // Уведомление для преподавателя
        Notification teacherNotification = new Notification();
        teacherNotification.setUser(template.getTeacher().getUser());
        teacherNotification.setMessage(String.format(
            "Ваш шаблон консультации (%s, %s-%s) был
автоматически удален по истечении установленного срока",
            template.getDayOfWeek().getDisplayName(TextStyle.FULL
, new Locale("ru")),
            template.getStartTime(),
            template.getEndTime()
        ));
        notificationRepository.save(teacherNotification);
        templateRepository.delete(template);
    }
    @Transactional
    public void updateConsultations(Teacher teacher) {
        LocalDateTime now = LocalDateTime.now();
        // Проверяются и удаляются шаблоны с истекшим сроком действия
        List<ConsultationTemplate> expiredTemplates =
templateRepository.findByTeacherId(teacher.getId())
            .stream()
            .filter(t -> t.getExpirationDateTime() != null &&
t.getExpirationDateTime().isBefore(now))
            .toList();
        expiredTemplates.forEach(this::deleteTemplateNotifications);
        // Удаляются прошедшие консультации
        List<Consultation> oldConsultations =
consultationRepository.findAll()
            .stream()
            .filter(c ->

```

```

        LocalDateTime.of(c.getConsultationDate(),
c.getEndTime()).isBefore(now))
            .filter(c ->
c.getTeacher().getId().equals(teacher.getId()))
            .toList();
        oldConsultations.forEach(consultation -> {
            // Очистение связи с архивными записями перед удалением
            List<ConsultationArchive> archives =
consultationArchiveRepository.findByConsultationId(consultation.getId());
            archives.forEach(archive -> {
                archive.setConsultation(null);
                consultationArchiveRepository.save(archive);
            });
        });
        consultationRepository.deleteAll(oldConsultations);
        // Получение активных шаблонов преподавателя
        List<ConsultationTemplate> templates =
templateRepository.findByTeacherId(teacher.getId());
        for (ConsultationTemplate template : templates) {
            // Получение всех активных консультаций по этому шаблону
            List<Consultation> existConsultations =
consultationRepository.findByTemplateId(template.getId())
                .stream()
                .filter(t -> !t.getTemplate().getIsOneTime())
                .toList();
            // Проверка, есть ли уже будущие консультации по этому
шаблону
            boolean hasFutureConsultations =
existConsultations.stream()
                .anyMatch(c ->
LocalDateTime.of(c.getConsultationDate(), c.getEndTime()).isAfter(now));
            // Создание новой консультации, если нет будущих
            if (!hasFutureConsultations) {
                LocalDate nextDate = getNextAvDate(template, now);
                if (template.getIsBiWeekly()) {
                    LocalDate lastConsultationDate =
existConsultations.stream()
                        .map(Consultation::getConsultationDate)
                        .max(LocalDate::compareTo)
                        .orElse(null);
                    if (lastConsultationDate != null &&
!nextDate.isAfter(lastConsultationDate.plusWeeks(1))) {
                        nextDate = lastConsultationDate.plusWeeks(2);
                    }
                }
                Consultation newConsultation = new Consultation();
                newConsultation.setTemplate(template);
                newConsultation.setTeacher(teacher);
                newConsultation.setConsultationDate(nextDate);
                newConsultation.setStartTime(template.getStartTime());
;
                newConsultation.setEndTime(template.getEndTime());
                newConsultation.setRoom(template.getRoom());
                newConsultation.setMaxStudents(template.getMaxStudent
s());
                consultationRepository.save(newConsultation);
            }
            // Проверяется заполненность студентов
            for (Consultation consultation : existConsultations) {

```

					BKP 090301.06 25 81 01	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		


```

        if (consultation.getMaxStudents() != null) {
            long signupsCount =
consultationSignupRepository.countByConsultationId(consultation.getId());
            // Если консультация заполнена и еще не прошла по
времени
            if (signupsCount >= consultation.getMaxStudents())
&&
                LocalDateTime.of(consultation.getConsulta
tionDate(), consultation.getEndTime()).isAfter(now)) {
                    // Проверяется, есть ли уже следующая
консультация
                    LocalDate nextDate =
consultation.getConsultationDate().plusWeeks(template.getIsBiWeekly() ? 2
: 1);
                    boolean nextConsultationExists =
consultationRepository.existsConsultationDateStartTime(
                        teacher.getId(),
                        nextDate,
                        consultation.getStartTime()
                    );
                    if (!nextConsultationExists) {
                        Consultation newConsultation = new
Consultation();
                        newConsultation.setTemplate(template);
                        newConsultation.setTeacher(teacher);
                        newConsultation.setConsultationDate(nextD
ate);
                        newConsultation.setStartTime(consultation
.getStartTime());
                        newConsultation.setEndTime(consultation.g
etEndTime());
                        newConsultation.setRoom(consultation.getR
oom());
                        newConsultation.setMaxStudents(consultati
on.getMaxStudents());
                        consultationRepository.save(newConsultati
on);
                    }
                }
            }
        }
    }
}

```

Файл ConsultationSignupController.java

```
package com.university.consultations.controller;

import com.university.consultations.entity.Consultation;
import com.university.consultations.entity.ConsultationArchive;
import com.university.consultations.entity.ConsultationSignup;
import com.university.consultations.entity.Student;

import
com.university.consultations.repository.ConsultationArchiveRepository;
import
com.university.consultations.repository.ConsultationRepository;
```

```

import
com.university.consultations.repository.ConsultationSignupRepository;
import com.university.consultations.repository.StudentRepository;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.server.ResponseStatusException;
import java.sql.Timestamp;
import java.time.LocalDateTime;
import java.util.List;
import java.util.Map;
@RestController
@RequestMapping("/api/consultations")
public class ConsultationSignupController {
    // Репозитории для работы с данными
    private final StudentRepository studentRepository;
    private final ConsultationRepository consultationRepository;
    private final ConsultationSignupRepository signupRepository;
    private final ConsultationArchiveRepository archiveRepository;
    public ConsultationSignupController(StudentRepository
studentRepository,
                                     ConsultationRepository
consultationRepository,
                                     ConsultationSignupRepository,
                                     ConsultationArchiveRepository
archiveRepository) {
        this.studentRepository = studentRepository;
        this.consultationRepository =
consultationRepository;
        this.signupRepository = signupRepository;
        this.archiveRepository = archiveRepository;
    }
    @PostMapping("/register") // Запись студента на консультацию
    public ResponseEntity<?> registerForConsultation(
        @RequestBody RegisterRequest request,
        Authentication authentication) {
        try {
            Student student =
studentRepository.findByUser_Login(authentication.getName());
            if (student == null) {
                return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                    .body(Map.of("error", "Пользователь не
авторизован"));
            } // Проверка существования консультации
            Consultation consultation =
consultationRepository.findById(request.getConsultationId())
                .orElseThrow(() -> new ResponseStatusException(
                    HttpStatus.NOT_FOUND,
                    "Консультация не найдена"
                )); // Проверка дублирования записи
            if
(signupRepository.existsByConsultationIdStudentId(consultation.getId(),
student.getId())) {
                return ResponseEntity.badRequest()
                    .body(Map.of("error", "Вы уже записаны на эту

```

					BKP 090301.06 25 81 01	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

```

        консультацию"));
        } // Проверка доступных мест
        long signupsCount =
signupRepository.countByConsultationId(consultation.getId());
        if (signupsCount >= consultation.getMaxStudents()) {
            return ResponseEntity.badRequest()
                .body(Map.of("error", "Достигнуто
максимальное количество записей"));
        }
        // Создание новой записи
        ConsultationSignup signup = new ConsultationSignup();
        signup.setConsultation(consultation);
        signup.setStudent(student);
        signup.setReason(request.getReason());
        signup.setSignupTime(LocalDateTime.now());
        signupRepository.save(signup);
        // Создаание архивной записи
        ConsultationArchive archive = new ConsultationArchive();
        archive.setConsultation(consultation);
        archive.setStudent(student);
        archive.setAttended(null);
        archive.setFeedback(null);
        archive.setArchivedAt(Timestamp.valueOf(LocalDateTime.now
()));
        archive.setReason(request.getReason());
        archive.setConsultationDate(consultation.getConsultationD
ate());
        archive.setStartTime(consultation.getStartTime());
        archive.setEndTime(consultation.getEndTime());
        archive.setRoom(consultation.getRoom());
        archive.setTeacherId(consultation.getTeacher().getId());
        archiveRepository.save(archive);
        // Возвращение успешного ответа с данными о записи
        return ResponseEntity.ok(Map.of(
            "success", "Вы успешно записаны на консультацию",
            "signupId", signup.getId(),
            "consultationDate",
consultation.getConsultationDate().toString(),
            "teacherName",
consultation.getTeacher().getLastName() + " " +
consultation.getTeacher().getFirstName()
+ " " +
consultation.getTeacher().getPatronymic()
));
    } catch (Exception e) {
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body(Map.of("error", "Ошибка сервера: " +
e.getMessage()));
    }
} // Завершение консультации и архивирование данных
@PostMapping("/end-consultation/{consultationId}")
public ResponseEntity<?> endConsultation(@PathVariable Integer
consultationId) { // Поиск консультации
    Consultation consultation =
consultationRepository.findById(consultationId)
        .orElseThrow(() -> new
ResponseStatusException(HttpStatus.NOT_FOUND, "Консультация не найдена"));

```

					ВКР 090301.06 25 81 01	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

        //Обработка всех записей на эту консультацию
        List<ConsultationSignup> signups = signupRepository.
findByConsultationId(consultationId); // Обновление архивной записи
        for (ConsultationSignup signup : signups) {
            ConsultationArchive archive =
archiveRepository.findByConsultationIdAndStudentId(consultationId,
signup.getStudent().getId())
                .stream()
                .findFirst()
                .orElse(new ConsultationArchive());
            archive.setConsultation(consultation);
            archive.setStudent(signup.getStudent());
            archive.setAttended(true);
            archive.setFeedback("Добавить отзыв");
            archive.setReason(signup.getReason());
            archive.setConsultationDate(consultation.getConsultationD
ate());

            archive.setStartTime(consultation.getStartTime());
            archive.setEndTime(consultation.getEndTime());
            archive.setRoom(consultation.getRoom());
            archive.setTeacherId(consultation.getTeacher().getId());
            archive.setTeacher(consultation.getTeacher());
            archive.setArchivedAt(Timestamp.valueOf(LocalDateTime.now
()));

            archiveRepository.save(archive);
        } // Удаление временных записей
        signupRepository.deleteAll(signups);
        return ResponseEntity.ok(Map.of("success", "Консультация
завершена, записи архивированы"));
    }
    @Data // DTO для запроса на регистрацию
    @NoArgsConstructor
    @AllArgsConstructor
    public static class RegisterRequest {
        private Integer consultationId;// ID консультации
        private String reason; // Причина записи
    }
}

```

Файл AuthController.java

```

package com.university.consultations.controller;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
@Controller
public class AuthController {
    @GetMapping("/login") // Отображение страницы входа в систему
    public String showLoginPage() {
        return "login";
    }
    @GetMapping("/redirectByRole") // Перенаправление пользователя
    public String redirectByRole(Authentication authentication) {
        if (authentication != null &&
authentication.isAuthenticated()) { // Проверка ролей пользователя
            for (GrantedAuthority authority :
authentication.getAuthorities()) {

```

```

        if (authority.getAuthority().equals("ROLE_Преподаватель")) {
            return "redirect:/teacher/dashboard";
        } else if
(authority.getAuthority().equals("ROLE_Студент")) {
            return "redirect:/student/dashboard";
        } else if
(authority.getAuthority().equals("ROLE_Руководство")) {
            return "redirect:/management/dashboard";
        }
    }
}
return "redirect:/login?error=unknown_role"; // Случай
нераспознавания
    }
}

```

Файл MvcConfig.java

```

package com.university.consultations.config;

import org.springframework.context.annotation.Configuration;
import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class MvcConfig implements WebMvcConfigurer {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry)
    {
        registry.addResourceHandler("/uploads/**") // URL-шаблон, по
которому будут доступны ресурсы
                .addResourceLocations("file:uploads/"); // путь к
файлам на сервере
    }
}

```

