

Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

Курсовой проект

по дисциплине

«Проектирование систем автоматизированного проектирования»

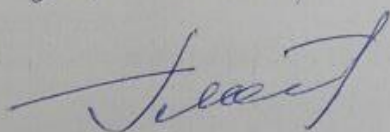
на тему «Проектирование подсистемы учебной САПР для автоматизации  
учета и записи консультаций на кафедре»

Направление подготовки 09.03.01 «Информатика и вычислительная техника»  
профиль «Системы автоматизированного проектирования»

21ВВ410

от

27.12.24



Выполнил ст. группы 21ВВ41

Нагорная Д. А.

Руководитель:

к.т.н. доцент Глотова Т. В.

Пенза 2024

Кафедра САПР

УТВЕРЖДАЮ

Зав. кафедрой

"09" 09 24 г.

## ЗАДАНИЕ

### на курсовое проектирование по курсу

Проектирование САПР

Студенту Нагорной Д.А. Группа 21ВВСТ

Тема проекта Проектирование подсистемы учебной САПР для автоматизации учёта и записи консультаций на кафедре.

Исходные данные (технические требования) на проектирование

- должна быть разработана диаграмма прецедентов подсистемы по стандарту UML;
- должна быть разработана структура разрабатываемой подсистемы;
- должно быть выполнено проектирование компонентов ИО и ПО (диаграммы коммуникации, классов, компонентов, развертывания и др.)

Подсистема предназначена для решения следующих задач:

- запись на консультацию (запись студента к преподавателю на консультацию);
- организация порядка;
- функционирования специализированного графического редактора;
- организации дружественного интерфейса с пользователем в виде системного меню, контекстно-зависимой, контекстно-независимой системы подсказок;
- оперативного управления специализированной базой данных.

Основные требования к программному обеспечению: объектно-ориентированный подход к проектированию и программированию, документируемость, модульность, соответствие стандартам интерфейса, соответствие принципу развития.

Язык программирования

Основные требования к информационному обеспечению: обеспечение целостности и непротиворечивости данных, эффективное управление БД.

## Объем работы по курсу

### 1. Расчетная часть

- 1.1 Анализ ТЗ
- 1.2 Анализ методов и алгоритмов проектирования
- 1.3 Разработка диаграмм прецедентов и последовательностей
- 1.4 Структура подсистемы:
- 1.5 Разработка структуры диалога
- 1.6 Информационное обеспечение (инфологическая, логическая и физическая модели данных);
- 1.7 Структура программного обеспечения:
- 1.8 Разработка ПО.

### 2. Графическая часть

структура подсистемы,

структура программного обеспечения (диаграммы классов, компонентов)

схема алгоритма

### 3. Экспериментальная часть

3.1 Разработка прототипа.

## Срок выполнения проекта по разделам

|   |                                  |                 |
|---|----------------------------------|-----------------|
| 1 | Анализ ТЗ                        | к 16.09.2024 г. |
| 2 | 1.2                              | к 23.09.2024 г. |
| 3 | 1.3, 1.4                         | к 30.09.2024 г. |
| 4 | 1.5                              | к 14.10.2024 г. |
| 5 | 1.6, 1.7                         | к 28.10.2024 г. |
| 6 | 1.8                              | к 18.11.2024 г. |
| 7 | 2 и 3                            | к 02.12.2024 г. |
| 8 | Оформление пояснительной записки | к 05.12.2024 г. |

Дата выдачи задания "9" сентября 2024 г.

Дата защиты проекта "9" декабря 2024 г.

Руководитель

Задание получил "9" сентября 2024 г.

Студент

Нагорная Д.А.

Пояснительная записка содержит 50 листов, 33 рисунка, 10 источников, 2 приложения.

ПРОЕКТИРОВАНИЕ УЧЕБНОЙ САПР ДЛЯ АВТОМАТИЗАЦИИ УЧЕТА И ЗАПИСИ КОНСУЛЬТАЦИЙ НА КАФЕДРЕ, STARUML, ARGOUML, FIGMA, JAVA, JAVAFX, MYSQL WORKBENCH.

Цель работы – изучение принципов объектно-ориентированного программирования на примере разработки пользовательского оконно-графического интерфейса, проектирование и разработка интерфейса пользователя для прикладных задач в системах автоматизированного проектирования, построение диаграмм UML, разработка специализированного графического редактора и проектирование конструкторской базы данных и разработка программы управления базой данных САПР.

Результатом разработки данного курсового проекта является специализированный графический редактор, включающий организацию дружественного интерфейса с пользователем в виде системного меню, контекстно-зависимой, контекстно-независимой системы подсказок. Разработка производилась с использованием масштабного стека технологий, к которым относятся: стандартизованный язык графического описания UML для моделирования систем и программного обеспечения StarUML и ArgoUML, графический редактор для проектирования интерфейса пользователя Figma, строго типизированный объектно-ориентированный язык программирования Java, платформа на основе Java для создания приложений с насыщенным графическим интерфейсом JavaFX.

|                  |             |                       |                |             |                                                                                             |                        |             |               |
|------------------|-------------|-----------------------|----------------|-------------|---------------------------------------------------------------------------------------------|------------------------|-------------|---------------|
|                  |             |                       |                |             | <i>ПГУ 4.09.03.01.06.001 ПЗ</i>                                                             |                        |             |               |
| <i>Изм.</i>      | <i>Лист</i> | <i>№ докум.</i>       | <i>Подпись</i> | <i>Дата</i> |                                                                                             |                        |             |               |
| <i>Разраб.</i>   |             | <i>Нагорная Д. А.</i> |                |             | <i>Проектирование учебной САПР для автоматизации учета и записи консультаций на кафедре</i> | <i>Лит.</i>            | <i>Лист</i> | <i>Листов</i> |
| <i>Провер.</i>   |             | <i>Глотова Т. В.</i>  |                |             |                                                                                             |                        | 4           | 1             |
| <i>Реценз.</i>   |             |                       |                |             |                                                                                             | <i>ФВТ, гр. 21BBC1</i> |             |               |
| <i>Н. контр.</i> |             |                       |                |             |                                                                                             |                        |             |               |
| <i>Утверд.</i>   |             |                       |                |             |                                                                                             |                        |             |               |

# Содержание

|                                                      |    |
|------------------------------------------------------|----|
| Введение.....                                        | 6  |
| 1 Расчетная часть.....                               | 7  |
| 1.1 Анализ ТЗ.....                                   | 7  |
| 1.2 Анализ методов и алгоритмов проектирования ..... | 7  |
| 1.3 Разработка диаграмм .....                        | 8  |
| 1.3.1 Разработка диаграммы прецедентов.....          | 9  |
| 1.3.1 Разработка диаграммы последовательностей.....  | 11 |
| 1.4 Структура подсистемы.....                        | 15 |
| 1.5 Разработка структуры диалога .....               | 17 |
| 1.6 Информационное обеспечение.....                  | 19 |
| 1.6.1 Инфологическая модель данных.....              | 19 |
| 1.6.2 Логическая модель данных.....                  | 20 |
| 1.6.3 Физическая модель данных .....                 | 21 |
| 1.7 Разработка прототипа подсистемы .....            | 22 |
| 2 Графическая часть.....                             | 28 |
| 2.1 Структура подсистемы.....                        | 28 |
| 2.2 Структура программного обеспечения.....          | 28 |
| 2.2.1 Диаграммы классов .....                        | 28 |
| 2.2.2 Диаграммы компонентов .....                    | 30 |
| 2.3 Схема алгоритма .....                            | 31 |
| 3 Разработка прототипа .....                         | 33 |
| 3.1 Отладка прототипа подсистемы .....               | 33 |
| Заключение .....                                     | 37 |
| Список используемой литературы .....                 | 38 |
| Приложение А .....                                   | 39 |
| Приложение Б.....                                    | 45 |

## **Введение**

Проектирование систем автоматизированного проектирования (САПР) представляет собой важный этап в разработке программного обеспечения, который определяет структуру, функциональность и взаимодействие различных компонентов системы. В условиях современного мира, где технологии развиваются стремительными темпами, создание эффективных и интуитивно понятных программных решений становится неотъемлемой частью успешной деятельности в любой области.

Одной из ключевых задач проектирования является создание удобного и функционального интерфейса, который позволяет пользователю легко взаимодействовать с системой. Графические интерфейсы играют важную роль в восприятии и использовании программного обеспечения, так как они обеспечивают визуальное представление данных и позволяют пользователю выполнять необходимые операции с минимальными усилиями.

Процесс включает в себя проектирование и разработку интерфейса пользователя для прикладных задач в САПР, построение диаграмм UML, проектирование программных систем, проектирование и разработка специализированного графического редактора для заданной предметной области, проектирование базы данных. Данный процесс позволяет визуализировать структуру системы, определить взаимодействия между компонентами и проанализировать сценарии использования.

В рамках данной курсовой работы в конечном итоге был разработан специализированный графический редактор на языке Java с использованием библиотеки JavaFX для создания графического интерфейса, предназначенная для записи студентов на консультации к преподавателям. Эта программа демонстрирует применение объектно-ориентированного подхода и иллюстрирует важность проектирования в создании интуитивно понятных и удобных для пользователя решений.

# **1 Расчетная часть**

Данный раздел курсовой работы предназначен для комплексного проектирования подсистемы САПР, включая анализ требований, разработку моделей данных и создание прототипа программного обеспечения.

## **1.1 Анализ ТЗ**

Техническое задание определяет основные задачи, которые должны быть решены в процессе проектирования подсистемы САПР. К ним относится проектирование, в которой необходимо разработать диаграмму прецедентов и последовательностей системы по стандарту UML, что позволит визуализировать взаимодействие пользователей с системой и определить ключевые функциональные требования. Затем определяется структура подсистемы, включая определение компонентов и их взаимодействия. Далее необходимо спроектировать дружественный интерфейс пользователя, включая системное меню и контекстно-независимые подсказки, что повысит удобство работы с системой. Вскоре подсистема должна обеспечивать функционирование графического редактора, который будет использоваться для проектирования и визуализации данных, что является важным этапом в разработке. В конечном итоге будет спроектирована база данных, что также является ключевым аспектом в области разработки.

## **1.2 Анализ методов и алгоритмов проектирования**

Методы и алгоритмы проектирования могут быть применены для разработки подсистемы автоматизированного проектирования (САПР). Правильный выбор методов и алгоритмов является ключевым фактором для успешной реализации проекта, так как они определяют структуру, функциональность и производительность системы.

Основным методом проектирования является объектно-ориентированное проектирование. Объектно-ориентированный подход наиболее естественно соответствует реальному процессу разработки систем и не только

программных, который является итеративным и может потребовать внести изменения в уже разработанные и отлаженные компоненты системы.

Объектно-ориентированный анализ направлен на создание моделей, более близких к реальности, с использованием объектно-ориентированного подхода; это методология, при которой требования формируются на основе понятий классов и объектов, составляющих словарь предметной области. На результатах ООА формируются модели, на которых основывается OOD.

Объектно-ориентированное проектирование – это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления как логической и физической, так статической и динамической моделей проектируемой системы. Именно поддержка объектно-ориентированной декомпозиции отличает OOD от структурного проектирования. OOD создает основу для окончательной реализации системы с использованием методологии OOP.

Алгоритмы проектирования представляют собой обширное описание методов и подходов, которые обеспечивают функциональность и удобство использования системы. Эти алгоритмы не только помогают в разработке качественного программного обеспечения, но и способствуют созданию эффективных и интуитивно понятных пользовательских интерфейсов, что в конечном итоге повышает удовлетворенность пользователей.

### **1.3 Разработка диаграмм**

Важным этапом в процессе проектирования систем, включая автоматизированные системы проектирования, является разработка UML диаграмм. UML это графический язык для специфицирования создания визуализации и документирования систем, в которых большая роль принадлежит программному обеспечению. Диаграммы помогают визуализировать структуру, поведение и взаимодействие компонентов системы, что облегчает понимание и коммуникацию среди участников данного процесса. В данном курсовом проекте представлены две диаграммы: диаграмма прецедентов и диаграмма последовательностей.

### 1.3.1 Разработка диаграммы прецедентов

Диаграмма прецедентов (Use Case Diagram) или вариантов использования показывает совокупность прецедентов и действующих лиц (Actor), а также отношения между ними. Прецеденты – это независимое от реализации высокоуровневое представление того, что пользователь ожидает от системы, т.е. описание функциональности системы. Действующее лицо – это все, что взаимодействует с создаваемой системой. Варианты использования и действующие лица определяют сферу применения создаваемой системы. При этом прецеденты описывают все то, что происходит внутри системы, а действующие лица – то, что происходит снаружи. Эта диаграмма помогает визуализировать взаимодействие пользователей с системой и выявить ключевые требования к функциональности. Она служит основой для проектирования и разработки, обеспечивая четкое понимание задач, которые необходимо реализовать.

Для каждого участника системы была разработана своя диаграмма прецедентов.

Диаграмма прецедентов для администратора системы представлена на рисунке 1.



Рисунок 1 – Диаграмма прецедентов для администратора системы

Администратор системы выполняет следующие задачи: создает учетные записи для преподавателей и студентов и управляет системой. Вариантами использования являются:

- создание учетной записи преподавателя/студента: ввод данных преподавателя/студента (ФИО, логин, пароль, факультет, кафедра, группа и т.д.) в систему;
- удаление учетной записи: удаление учетной записи пользователя из системы.

Диаграмма прецедентов для преподавателя представлена на рисунке 2.

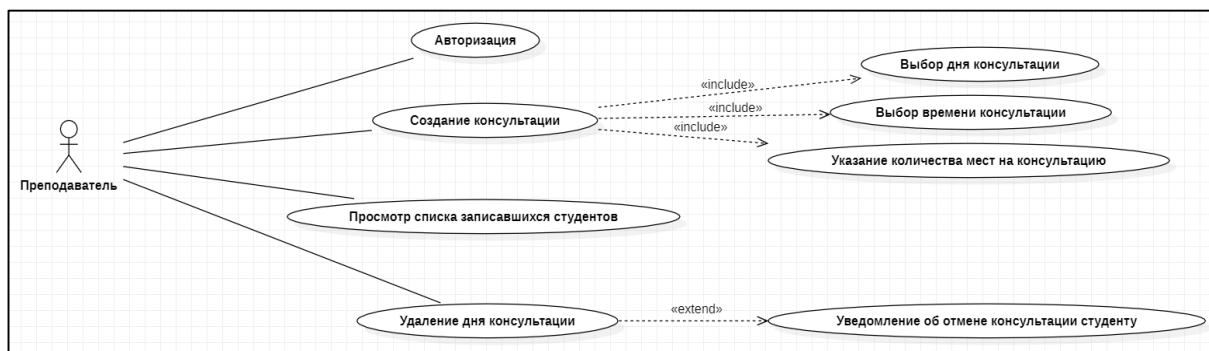


Рисунок 2 – Диаграмма прецедентов для преподавателя

Преподаватель выполняет следующие задачи: проводит консультации, создает дни консультаций и управляет списком студентов. Вариантами использования являются:

- авторизация: ввод логина и пароля для входа в систему;
- создание консультации: выбор дня консультации, выбор времени консультации, указание количества мест на консультацию;
- просмотр списка записавшихся студентов (просмотр списка студентов, записавшихся на консультацию);
- удаление дня консультации (удаление созданной консультации): уведомление об отмене консультации студенту (отправление уведомления студентам, записанным на отмененную консультацию).

Диаграмма прецедентов для студента представлена на рисунке 3.

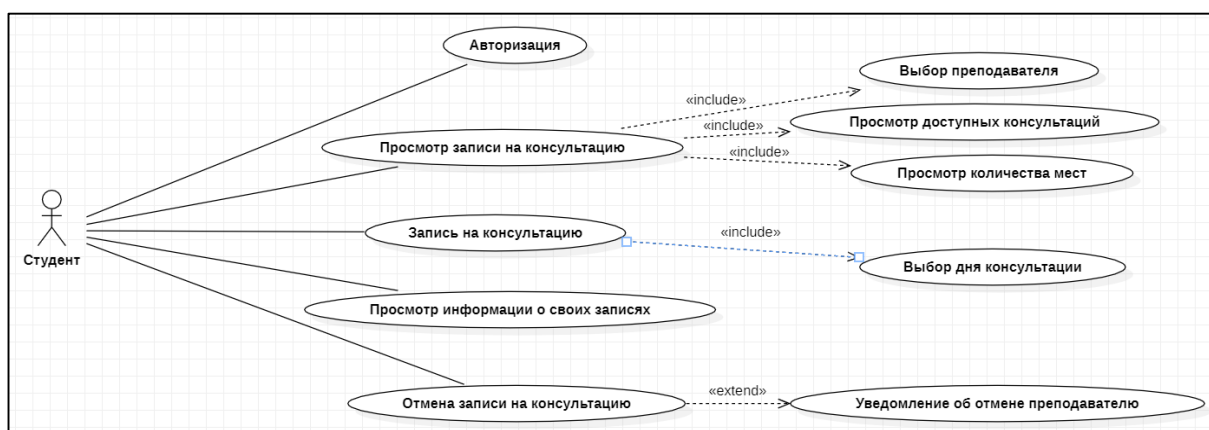


Рисунок 3 – Диаграмма прецедентов для студента

Студент выполняет следующие задачи: записывается на консультацию, отменяет запись, получает информацию о консультациях. Вариантами использования являются:

- авторизация: ввод логина и пароля для входа в систему;
- просмотр расписания консультаций у преподавателя: выбор преподавателя (выбор преподавателя из списка по поиску), просмотр доступных консультаций (просмотр списка консультаций преподавателя, которые доступны для записи), просмотр количества мест (просмотр количества мест на каждой консультации);
- запись на консультацию: выбор дня консультации (выбор дня консультации из списка);
- просмотр информации о своих записях (просмотр списка своих записей на консультации);
- отмена записи на консультацию (отмена записи на консультацию): уведомление об отмене преподавателю (отправление уведомления преподавателю на отмененную запись).

### **1.3.1 Разработка диаграммы последовательностей**

Диаграммой последовательностей (Sequence Diagram) называется диаграмма взаимодействия, акцентирующая внимание на временной упорядоченности сообщений. Графически такая диаграмма представляет собой таблицу, объекты в которой располагаются вдоль оси X, а сообщения в порядке возрастания времени – вдоль оси Y.

На диаграмме последовательностей показаны линии жизни объектов и фокус управления (активизации объекта). Обычно для каждого варианта использования создаются несколько диаграмм последовательностей. Одна для варианта сценария без ошибок, другие отражают ход событий в альтернативных потоках. Отправление сообщений, которые могут обозначать событие или вызовы операций показываются горизонтальными стрелками с обозначением синхронизации. Линия, обозначающая посылку сообщения, проводится от вертикали клиента к вертикали сервера. Первое сообщение - на

самом высоком уровне, второе ниже, и т.д. Сообщение на диаграмме показывает, что один объект вызывает функцию другого. В дальнейшем, когда будут определены операции классов, каждое сообщение станет операцией.

Диаграмма последовательностей для администратора системы приведена на рисунке 4.



Рисунок 4 – Диаграмма последовательностей для администратора системы

Диаграмма показывает последовательность взаимодействия между администратором системы и базой данных. Для создания и удаления учетной записи администратор системы отправляет запрос на создание и удаления соответственно в базу данных, после чего из базы данных к нему отправляется ответ.

Первая диаграмма последовательностей для преподавателя приведена на рисунке 5.

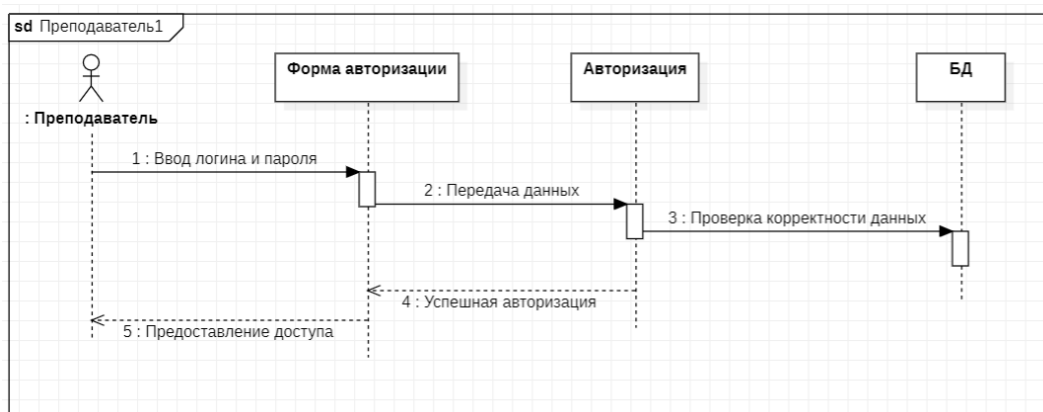


Рисунок 5 – Диаграмма последовательностей для преподавателя

Диаграмма показывает последовательность взаимодействия между пользователем и системой, включая все ключевые этапы авторизации и

обработки данных. Для начала пользователь (преподаватель) вводит логин и пароль через форму авторизации, далее данные передаются на авторизацию, затем системе происходит проверка корректности данных, при этом если данные корректны, происходит успешная авторизация, в конечном итоге после успешной авторизации предоставляется доступ к системе.

Вторая диаграмма последовательностей для преподавателя приведена на рисунке 6.



Рисунок 6 – Диаграмма последовательностей для преподавателя

На диаграмме изображены взаимодействия между преподавателем, процессом консультации и базой данных (БД). Диаграмма наглядно показывает, как проходит процесс управления консультациями, включая ввод данных, их обработку и возможность удаления. Для начала преподаватель инициирует процесс, вводя данные, затем система сохраняет данные в БД, далее идет подтверждение успешного создания консультации, вскоре преподаватель может просмотреть информацию, затем система предоставляет список студентов, записавшихся на консультацию, далее преподаватель видит список студентов, при этом преподаватель может удалить консультацию, вскоре система удаляет информацию из БД, в конечном итоге происходит подтверждение удаления.

Первая диаграмма последовательностей для студента приведена на рисунке 7.

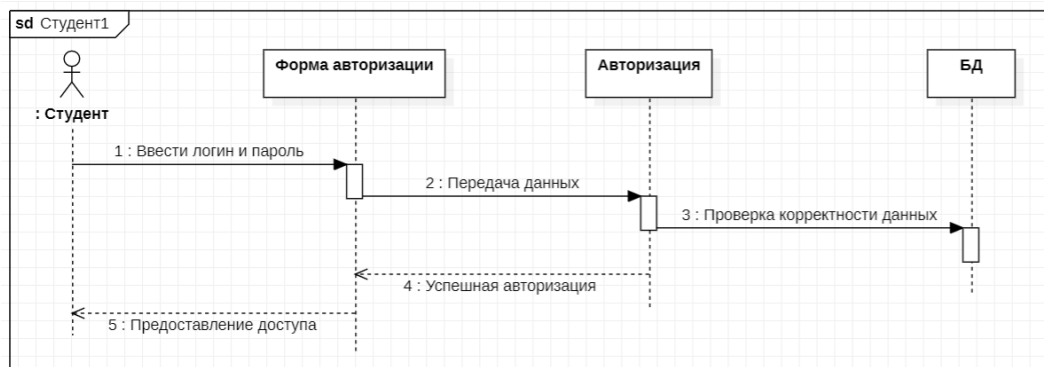


Рисунок 7 – Диаграмма последовательностей для студента

Диаграмма показывает последовательность взаимодействия между пользователем и системой, включая все ключевые этапы авторизации и обработки данных. Для начала пользователь (студент) вводит логин и пароль через форму авторизации, далее данные передаются на авторизацию, затем системе происходит проверка корректности данных, при этом если данные корректны, происходит успешная авторизация, в конечном итоге после успешной авторизации предоставляется доступ к системе.

Вторая диаграмма последовательностей для студента приведена на рисунке 8.

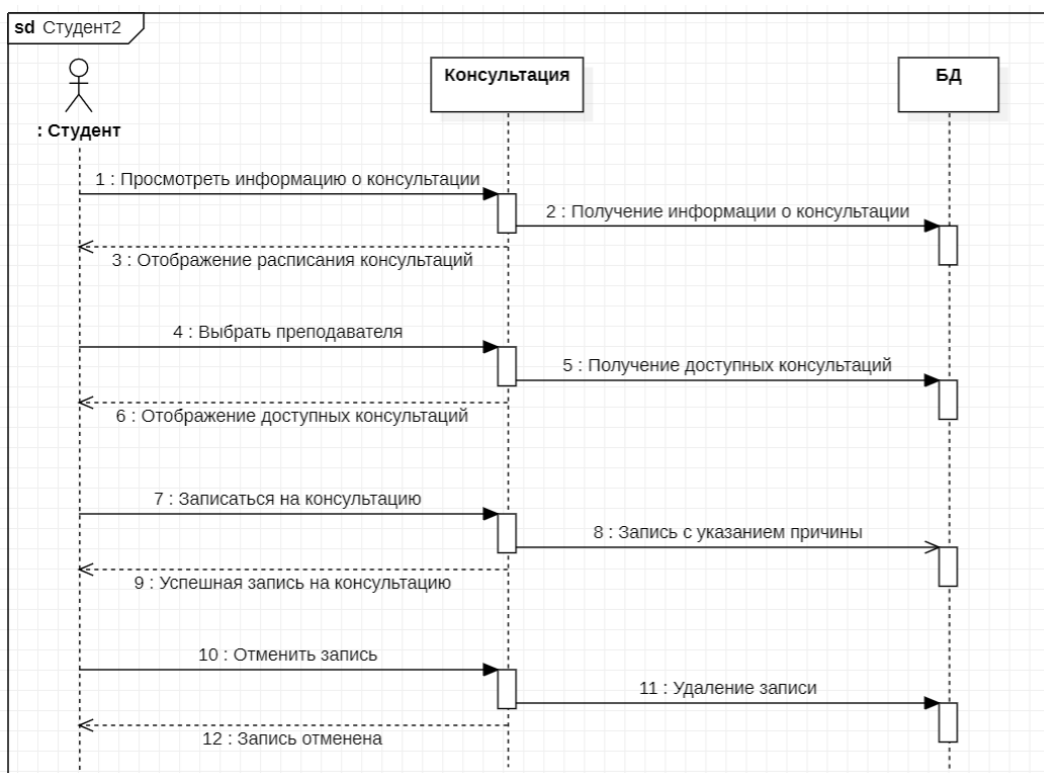


Рисунок 8 – Диаграмма последовательностей для студента

На диаграмме изображены взаимодействия между студентом, системой консультаций и базой данных (БД). Диаграмма помогает понять, как происходит процесс записи на консультацию и взаимодействие между пользователем и системой. Для начала студент инициирует процесс, запрашивая информацию, потом отвечает на запрос студента, следующий шаг – система показывает доступные консультации, в конце концов студент выбирает, к какому преподавателю хочет записаться, далее система предоставляет список доступных консультаций по выбранному преподавателю, затем система снова показывает доступные временные слоты, вскоре студент делает запись, далее система фиксирует запись, включая причину обращения, затем студент получает подтверждение, вскоре студент может отменить свою запись, при этом система удаляет запись из базы данных, в конце концов студент получает подтверждение об отмене.

#### 1.4 Структура подсистемы

Структура подсистемы включает несколько модулей: интерфейс, графический редактор, запись и учет, СУБД, справка, БД (рисунок 9). Она позволяет создавать, редактировать и сохранять консультации в интерактивном режиме.

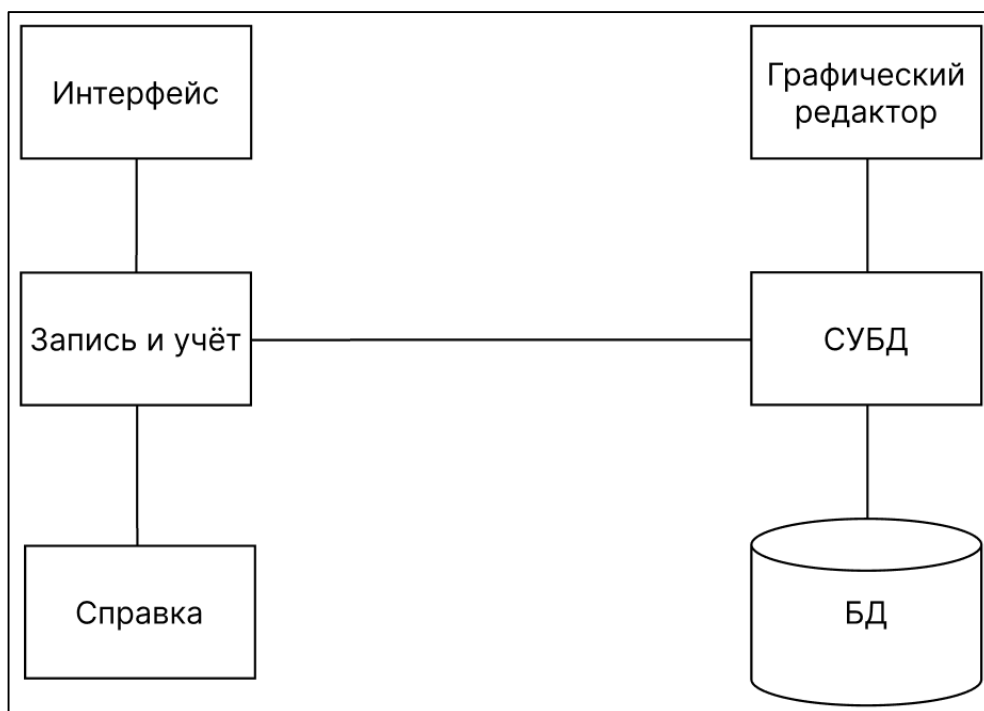


Рисунок 9 – Структура подсистемы

Интерфейс служит связующим звеном между пользователем и системой, позволяя взаимодействовать с модулем записи и учета, который фиксирует все действия и управляет записями консультаций. Этот модуль, в свою очередь, предоставляет пользователю доступ к справке, содержащей инструкции и советы по использованию системы, а также взаимодействует с системой управления базами данных (СУБД), обеспечивая надежное хранение и обработку данных. Графический редактор, также связанный с СУБД, позволяет пользователям создавать и редактировать визуальные элементы консультаций, в то время как сама СУБД обеспечивает целостность и доступность информации, хранящейся в базе данных (БД), что гарантирует структурированное и безопасное хранение всех материалов для дальнейшего доступа и анализа.

На рисунке 10 представлено развертывание модуля «Запись».

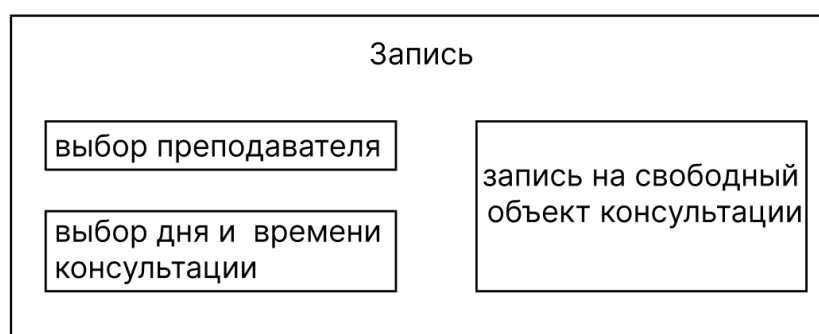


Рисунок 10 – Модуль «Запись»

На рисунке 11 представлено развертывание модуля «Учет».

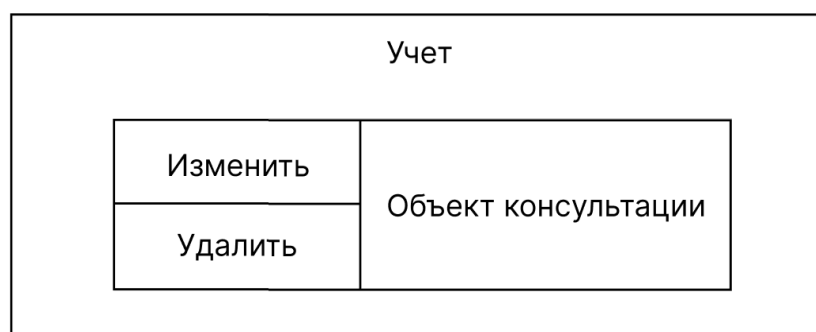


Рисунок 11 – Модуль «Учет»

Структура подсистемы включает несколько взаимосвязанных модулей, которые обеспечивают эффективное создание, редактирование и сохранение консультаций.

## 1.5 Разработка структуры диалога

Диалог в САПР является методом решения слабо формализованных задач. При этом разделение функций между человеком и ЭВМ следующее: человек ставит задачу, а система предоставляет средства для решения подзадач. Производится совместное решение подзадач, окончательное объединение результатов, а принятие проектных решений остается за человеком. Диалоговой системой (подсистемой) называется система (подсистема), обеспечивающая функционирование в режиме диалога. Диалоговый режим представляет собой способ организации вычислительного процесса, основанный на диалоговом взаимодействии пользователей и вычислительной машиной (системой).

Основное графическое представление структуры диалога – диаграмма состояний (граф диалога) – ориентированный взвешенный граф. Каждая вершина графа соответствует определенному состоянию диалога, а дуга определяет смену состояния. Веса переходов характеризуют содержание диалога: входные и выходные сообщения, исполняемые программы и программные условия.

Тип диалога характеризует структурные и лингвистические средства для оформления диалоговых обменов. Рассматриваются следующие типы диалога: абстрактный, конкретный, технической реализации. Каждый из этих типов имеет свои особенности и применяется в зависимости от контекста общения и целей взаимодействия.

Сценарий диалога – детальное описание структуры и содержания диалога, достаточное для его реализации автоматическим путем. Явное выделение сценария позволяет контролировать допустимые последовательности состояний, локализовать изменения структуры диалога, упростить разработку и отладку диалоговых программ, обеспечивает независимое тестирование диалога. Сценарий диалога кроме описания структуры включает в себя информационную и операционную модели диалога.

Граф диалога для преподавателя представлен на рисунке 12.

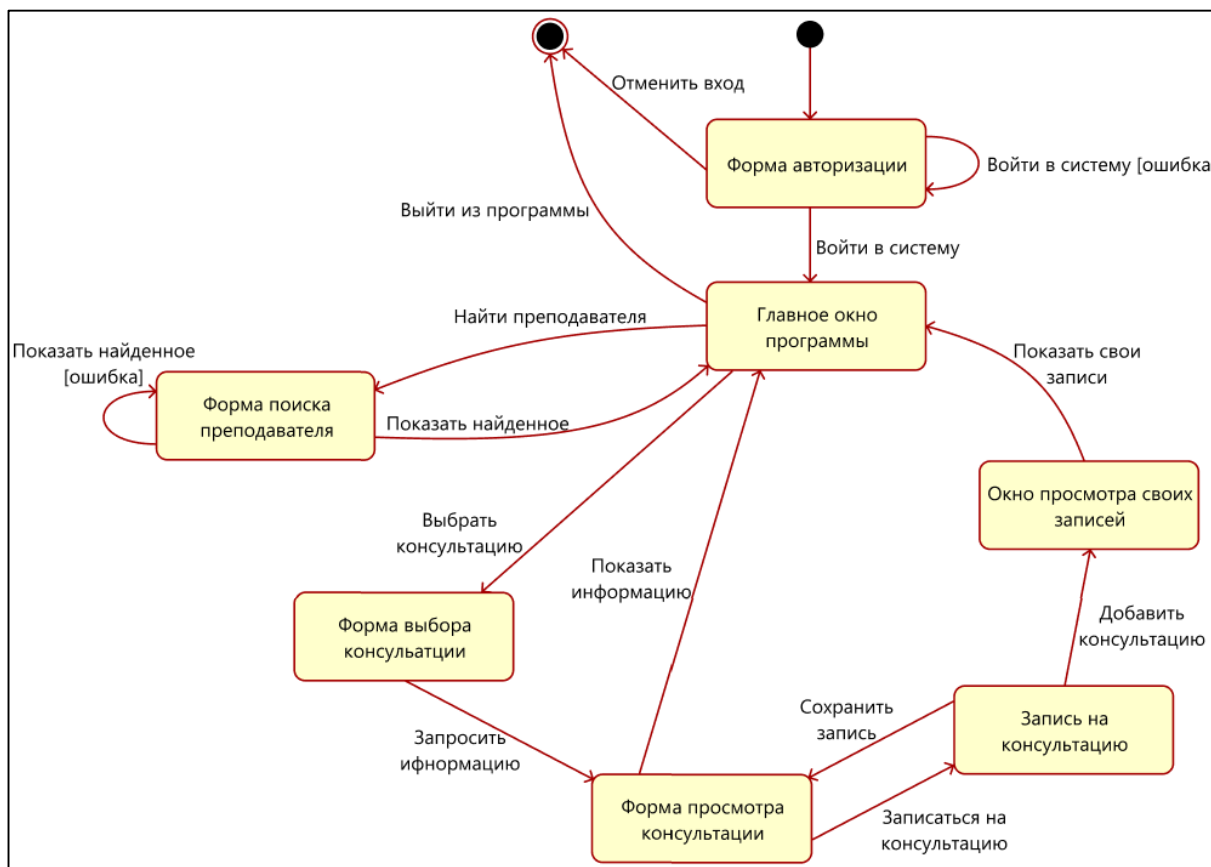


Рисунок 12 – Граф диалога для преподавателя

Граф диалога для студента представлен на рисунке 13.

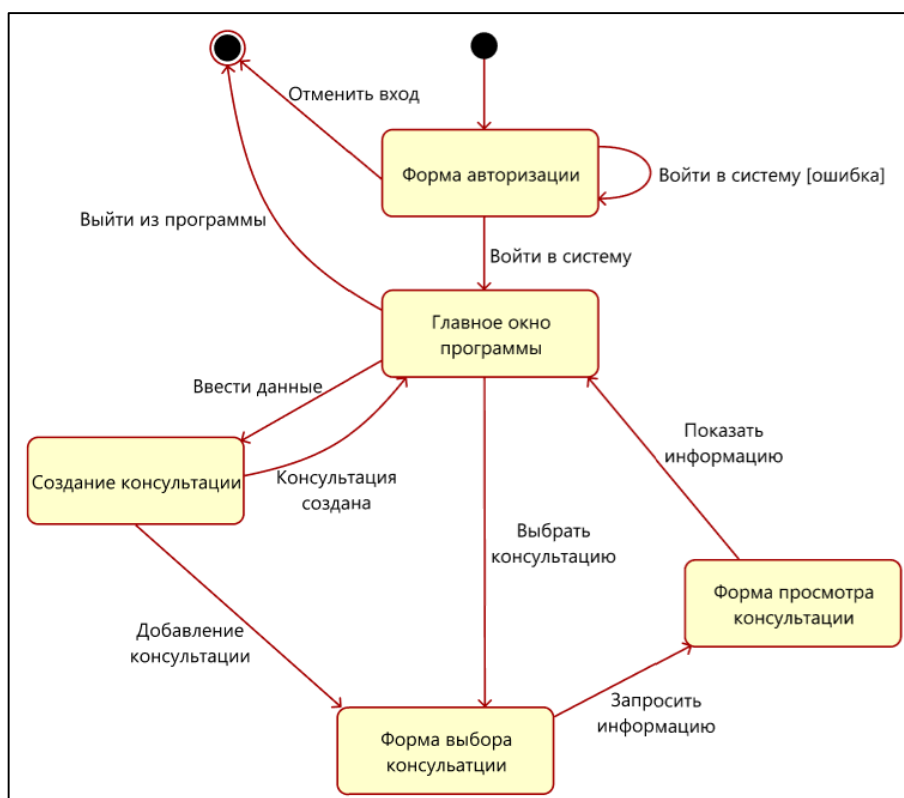


Рисунок 13 – Граф диалога для студента

## 1.6 Информационное обеспечение

Данный раздел охватывает процессы сбора, обработки, хранения и представления данных, необходимых для поддержки принятия решений и эффективного функционирования системы.

### 1.6.1 Инфологическая модель данных

Инфологическая модель данных – это описание данных и предметной области, не зависящее от особенностей системы управления базами данных (СУБД), на которой впоследствии будет создаваться база данных. На этом уровне выделяются основные сущности предметной области и отношения между ними. Эта модель служит основой для проектирования более детализированных логических и физических моделей данных. Правильное определение сущностей и их взаимосвязей на этом этапе критически важно для успешной реализации всей системы управления данными.

На рисунке 14 разработана модель предметной области в виде диаграммы «сущность-связь».

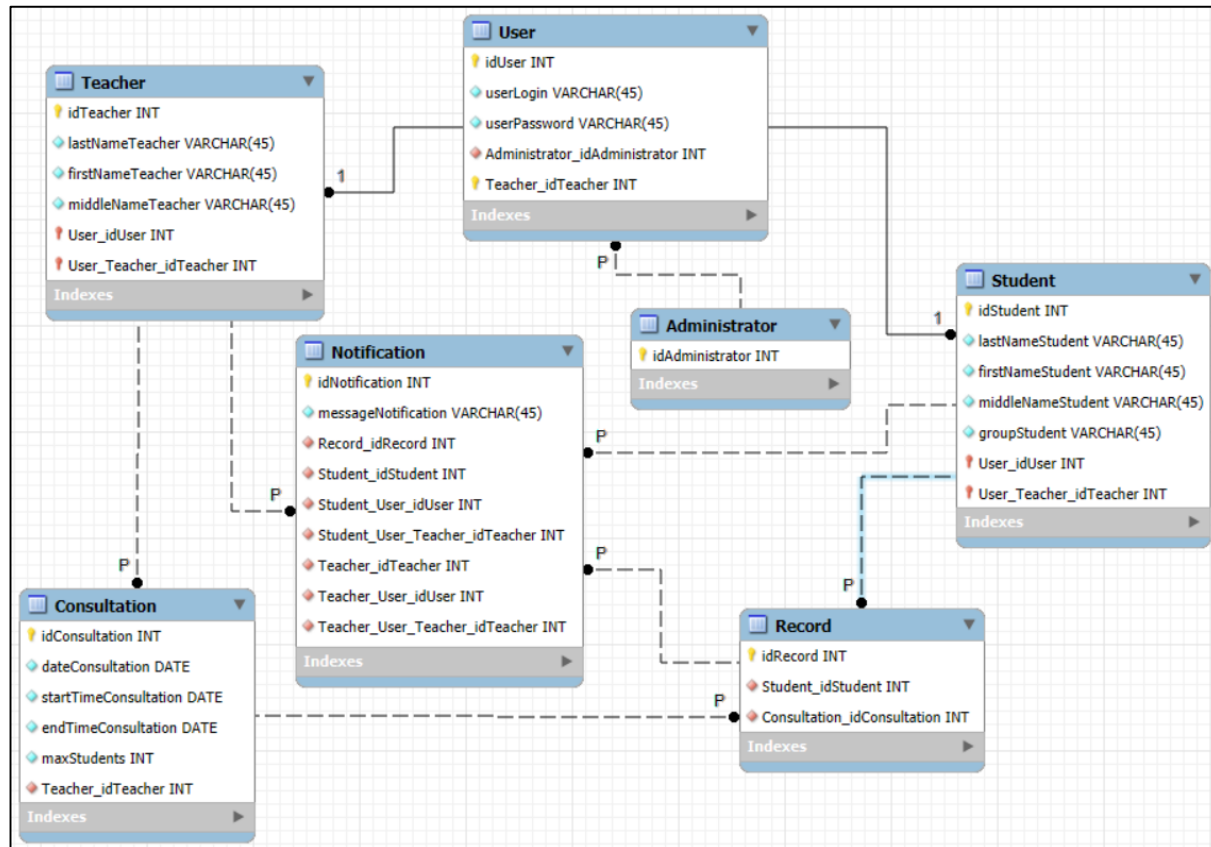


Рисунок 14 - Инфологическая модель данных IDEF1X

Для данной модели рассматриваются следующие связи:

- связь Administrator - User – неидентифицирующая связь 1:n;
- связь Teacher - User – идентифицирующая связь 1:1;
- связь Student - User – идентифицирующая связь 1:1;
- связь Teacher - Notification – неидентифицирующая связь 1:1;
- связь Student - Notification – неидентифицирующая связь 1:1;
- связь Teacher - Consultation – неидентифицирующая связь 1:n;
- связь Student - Record – неидентифицирующая связь 1:n;
- связь Consultation - Record – неидентифицирующая связь 1:n.

### 1.6.2 Логическая модель данных

Логическое проектирование заключается в определении числа и структуры таблиц, формировании запросов к БД, определении типов отчетных документов, разработке алгоритмов обработки информации, создании форм для ввода и редактирования данных в базе и решении других задач. Решение задач логического проектирования БД в основном определяется спецификой задач предметной области. Генерация SQL кода приведена в Приложении А.

Логическая модель данных представлена на рисунке 15.

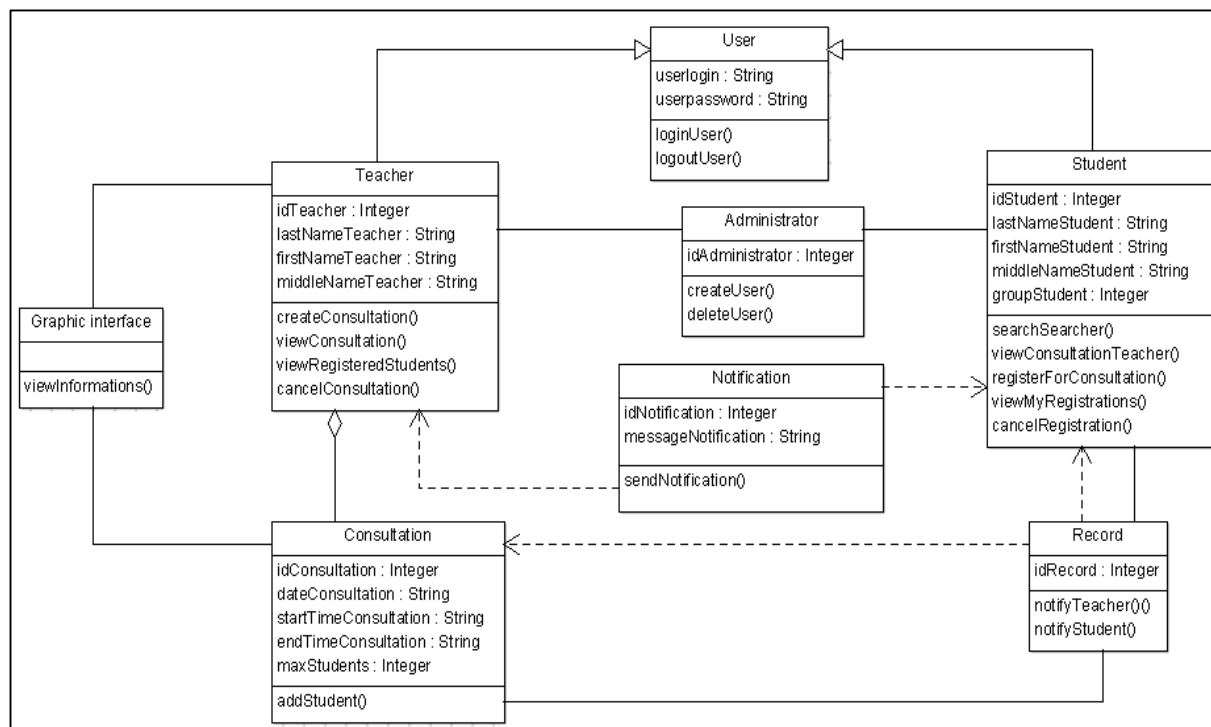


Рисунок 15 – Логическая модель данных

### 1.6.3 Физическая модель данных

Физическая модель данных – это детальное представление структуры данных, которое описывает, как данные будут храниться в базе данных. Она включает в себя таблицы, поля (атрибуты), типы данных, ограничения, индексы и связи между таблицами. Физическая модель является основой для реализации базы данных и помогает обеспечить целостность данных, оптимизацию запросов и эффективное управление данными. Физическая модель показывает:

1. Структуру хранения данных:

- как организованы данные о студентах, преподавателях, консультациях и записях;
- какие таблицы будут созданы в базе данных и какие поля они будут содержать.

2. Связи между сущностями:

- как студенты связаны с консультациями (через записи);
- как преподаватели связаны с консультациями;
- как записи связывают студентов с конкретными консультациями и преподавателями.

3. Ограничения и уникальность:

- уникальность ФИО студентов и преподавателей;
- ограничения на максимальное количество студентов на одной консультации.

В специализированном графическом редакторе будут выделены следующие сущности и атрибуты:

1. Таблица Students:

- id (PRIMARY KEY, уникальный идентификатор студента);
- group (группа студента);
- fullName (ФИО студента).

2. Таблица Consultations:

- id (PRIMARY KEY, уникальный идентификатор консультации);

- date (дата и время консультации).
- maxStudents (максимальное количество студентов).
- teacher\_id (FOREIGN KEY, идентификатор преподавателя).

### 3. Таблица Teachers:

- id (PRIMARY KEY, уникальный идентификатор преподавателя);
- name (ФИО преподавателя).

### 4. Таблица MyRecords:

- id (PRIMARY KEY, уникальный идентификатор записи);
- teacher\_id (FOREIGN KEY, идентификатор преподавателя);
- consultation\_id (FOREIGN KEY, идентификатор консультации);
- student\_id (FOREIGN KEY, идентификатор студента).

Между таблицами будут представлены следующие связи:

- связь Teachers (1) – Consultations (n): один преподаватель может иметь много консультаций;
- связь Consultations (1) – MyRecords (n): одна консультация может иметь много записей;
- связь Students (1) – MyRecords (n): один студент может иметь много записей.

## 1.7 Разработка прототипа подсистемы

В рамках проекта по разработке системы управления записями студентов на консультации преподавателей был создан графический интерфейс, который был тщательно спроектирован с использованием Figma. Этот этап включал в себя создание макетов, которые служат основой для визуального представления функционала и взаимодействия пользователей с системой. ключевые аспекты разработки прототипа подсистемы.

Ниже приведено проектирование макета (прототипа) подсистемы.

Для того, чтобы записаться на консультацию, необходимо выбрать объект – преподаватель. Для этого в поле ввода нужно ввести его ФИО (рисунок 16).

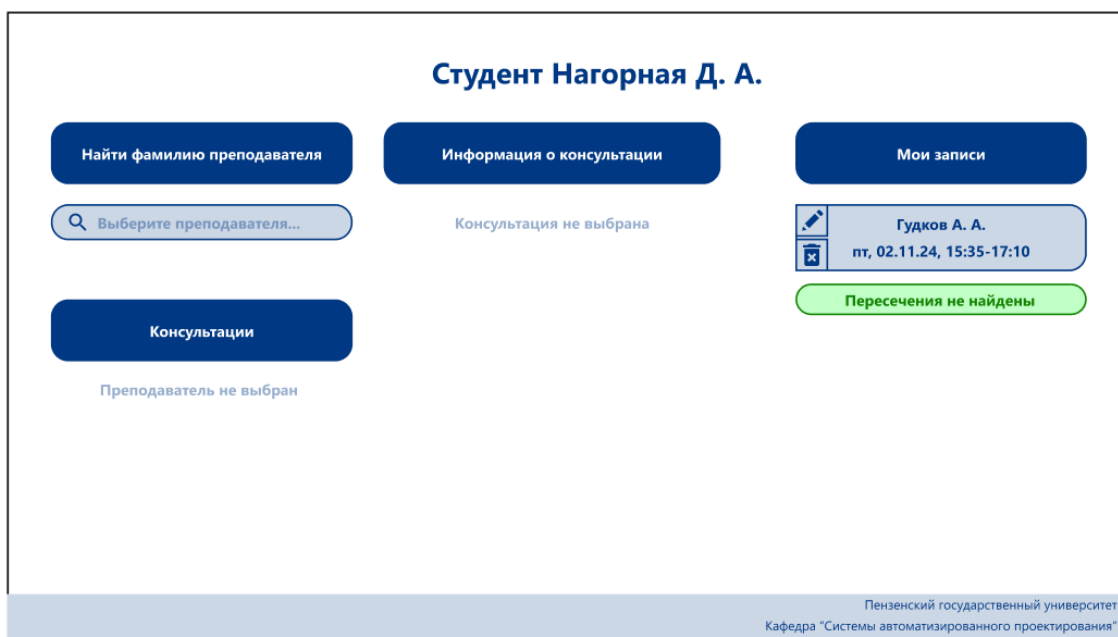


Рисунок 16 – Главное окно интерфейса

После того, как преподаватель был найден, выводится список его консультаций. Необходимо выбрать один день из консультаций (рисунок 17).

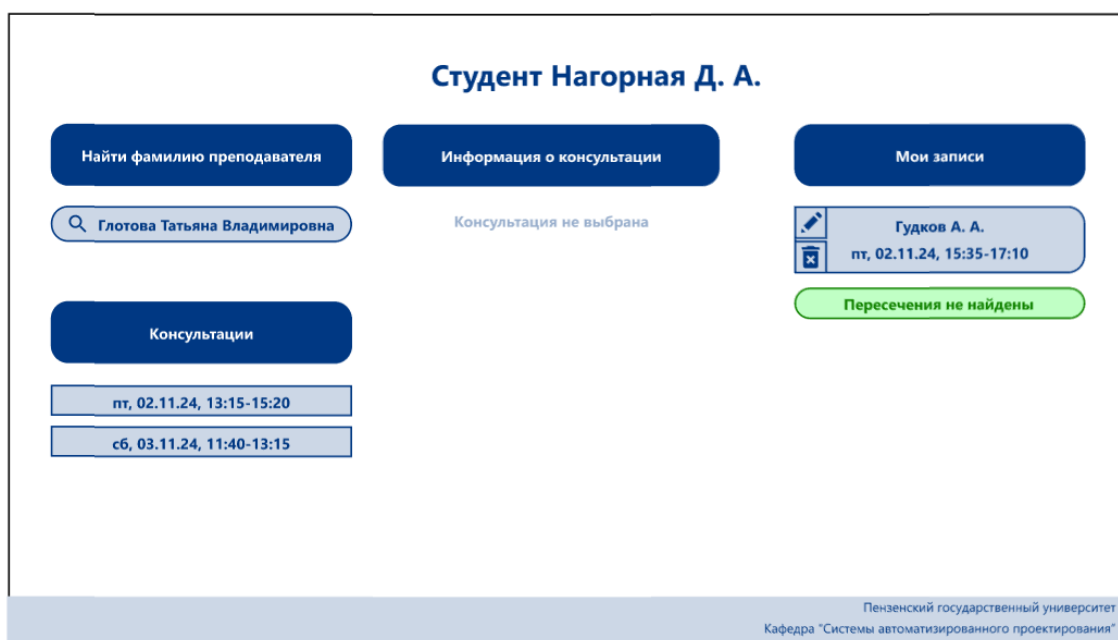


Рисунок 17 – Выбор преподавателя

После того, как выбран день и время консультации выводится список записавшихся студентов. Для записи необходимо выбрать свободный объект – консультации (рисунок 18).

**Студент Нагорная Д. А.**

Найти фамилию преподавателя

Консультации

пт, 02.11.24, 13:15-15:20

сб, 03.11.24, 11:40-13:15

Информация о консультации

| № | Группа | Фамилия       | Запись                     |
|---|--------|---------------|----------------------------|
| 1 | 21BBS1 | Иванов Д. С.  | Занято                     |
| 2 | 20BBS1 | Козлова Е. В. | Занято                     |
| 3 | 21BBS1 | Петров С. А.  | Занято                     |
| 4 | -      | -             | <a href="#">Записаться</a> |
| 5 | -      | -             | <a href="#">Записаться</a> |

Мои записи

✎ ✕

Гудков А. А.  
пт, 02.11.24, 15:35-17:10

Пересечения не найдены

Пензенский государственный университет  
 Кафедра "Системы автоматизированного проектирования"

Рисунок 18 – Выбор консультации и вывод списка

После записи на консультацию выводится список консультаций студента, на которые он записался. Нужно учитывать, чтобы не было пересечений, как приведено ниже. Чтобы это исправить, можно, к примеру, изменить запись (рисунок 19).

**Студент Нагорная Д. А.**

Найти фамилию преподавателя

Консультации

пт, 02.11.24, 13:15-15:20

сб, 03.11.24, 11:40-13:15

Информация о консультации

| № | Группа | Фамилия        | Запись                     |
|---|--------|----------------|----------------------------|
| 1 | 21BBS1 | Иванов Д. С.   | Занято                     |
| 2 | 20BBS1 | Козлова Е. В.  | Занято                     |
| 3 | 21BBS1 | Петров С. А.   | Занято                     |
| 4 | 21BBS1 | Нагорная Д. А. | Занято                     |
| 5 | -      | -              | <a href="#">Записаться</a> |

Мои записи

✎ ✕

Гудков А. А.  
пт, 02.11.24, 13:15-15:20

✎ ✕

Глотова Т. В.  
пт, 02.11.24, 13:15-15:20

Найдены пересечения!

Пензенский государственный университет  
 Кафедра "Системы автоматизированного проектирования"

Рисунок 19 – Запись на консультации и вывод предупреждения

После того, как студент нажал на кнопку изменения записи необходимо так же выбрать свободное окошко и записаться на консультацию. Но перед этим нужно выбрать другой день консультации (рисунок 20).

**Студент Нагорная Д. А.**

**Найти фамилию преподавателя**

🔍 Глотова Татьяна Владимировна

**Информация о консультации**

| № | Группа | Фамилия        | Запись                     |
|---|--------|----------------|----------------------------|
| 1 | 21BBS1 | Иванов Д. С.   | Занято                     |
| 2 | 20BBS1 | Козлова Е. В.  | Занято                     |
| 3 | 21BBS1 | Петров С. А.   | Занято                     |
| 4 | 21BBS1 | Нагорная Д. А. | Занято                     |
| 5 | -      | -              | <a href="#">Записаться</a> |

**Мои записи**

✎ Гудков А. А.  
пт, 02.11.24, 13:15-15:20

**Выберите запись...**

Найдены пересечения!

**Консультации**

пт, 02.11.24, 13:15-15:20

сб, 03.11.24, 11:40-13:15

Пензенский государственный университет  
Кафедра "Системы автоматизированного проектирования"

Рисунок 20 – Изменение записи

Как видно ниже, такой день есть и существуют свободные окошки. Поэтому, можно записаться на консультацию (рисунок 21).

**Студент Нагорная Д. А.**

**Найти фамилию преподавателя**

🔍 Глотова Татьяна Владимировна

**Информация о консультации**

| № | Группа | Фамилия      | Запись                     |
|---|--------|--------------|----------------------------|
| 1 | 21BBS1 | Грачев П. С. | Занято                     |
| 2 | -      | -            | <a href="#">Записаться</a> |
| 3 | -      | -            | <a href="#">Записаться</a> |
| 4 | -      | -            | <a href="#">Записаться</a> |
| 5 | -      | -            | <a href="#">Записаться</a> |

**Мои записи**

✎ Гудков А. А.  
пт, 02.11.24, 13:15-15:20

**Выберите запись...**

Найдены пересечения!

**Консультации**

пт, 02.11.24, 13:15-15:20

сб, 03.11.24, 11:40-13:15

Пензенский государственный университет  
Кафедра "Системы автоматизированного проектирования"

Рисунок 21 – Переход на другую консультацию

После записи появляется новая запись у студента и пропадает предупреждение о том, что консультации пересекаются (рисунок 22).

**Студент Нагорная Д. А.**

Найти фамилию преподавателя

🔍 Глотова Татьяна Владимировна

Консультации

пт, 02.11.24, 13:15-15:20

сб, 03.11.24, 11:40-13:15

Информация о консультации

| № | Группа | Фамилия        | Запись                     |
|---|--------|----------------|----------------------------|
| 1 | 21BBC1 | Грачев П. С.   | Занято                     |
| 2 | 21BBC1 | Нагорная Д. А. | Занято                     |
| 3 | -      | -              | <a href="#">Записаться</a> |
| 4 | -      | -              | <a href="#">Записаться</a> |
| 5 | -      | -              | <a href="#">Записаться</a> |

Мои записи

✎ Гудков А. А.  
пт, 02.11.24, 13:15-15:20

✎ Глотова Т. В.  
сб, 03.11.24, 11:40-13:15

Пересечения не найдены

Пензенский государственный университет  
Кафедра "Системы автоматизированного проектирования"

Рисунок 22 – Запись на другую консультацию

В другом случае, если других консультаций не найдено или по какой-то другой причине, можно удалить консультацию (рисунок 23).

**Студент Нагорная Д. А.**

Найти фамилию преподавателя

🔍 Глотова Татьяна Владимировна

Консультации

пт, 02.11.24, 13:15-15:20

сб, 03.11.24, 11:40-13:15

Информация о консультации

| № | Группа | Фамилия        | Запись                     |
|---|--------|----------------|----------------------------|
| 1 | 21BBC1 | Грачев П. С.   | Занято                     |
| 2 | 21BBC1 | Нагорная Д. А. | Занято                     |
| 3 | -      | -              | <a href="#">Записаться</a> |
| 4 | -      | -              | <a href="#">Записаться</a> |
| 5 | -      | -              | <a href="#">Записаться</a> |

Мои записи

✎ Гудков А. А.  
пт, 02.11.24, 13:15-15:20

Удалить консультацию?  
**да**   **нет**

Пересечения не найдены

Пензенский государственный университет  
Кафедра "Системы автоматизированного проектирования"

Рисунок 23 – Удаление консультации

После удаления пропадает запись в записях студента (рисунок 24).

Студент Нагорная Д. А.

Найти фамилию преподавателя

Консультации

пт, 02.11.24, 13:15-15:20

сб, 03.11.24, 11:40-13:15

Информация о консультации

| № | Группа | Фамилия        | Запись                     |
|---|--------|----------------|----------------------------|
| 1 | 21BBS1 | Грачев П. С.   | Занято                     |
| 2 | 21BBS1 | Нагорная Д. А. | Занято                     |
| 3 | -      | -              | <a href="#">Записаться</a> |
| 4 | -      | -              | <a href="#">Записаться</a> |
| 5 | -      | -              | <a href="#">Записаться</a> |

Мои записи

Гудков А. А.

пт, 02.11.24, 13:15-15:20

Пересечения не найдены

Пензенский государственный университет

Кафедра "Системы автоматизированного проектирования"

Рисунок 24 – Вывод конечного списка консультаций студента

## **2 Графическая часть**

В графической части представлены чертежи: структура подсистемы и структура программного обеспечения.

### **2.1 Структура подсистемы**

Определен набор графических операций для заданной подсистемы проектирования. В данном случае выбраны графические операции для студента, которые приведены ниже:

1. Запись на консультацию:
  - выбор преподавателя;
  - выбор дня и времени консультации;
  - запись на свободный объект консультации.
2. Отмена записи на консультацию:
  - изменение записи на консультацию;
  - удаление записи на консультацию.
3. Изменение записи на консультацию (выполняется 1 пункт):
  - выбор преподавателя;
  - выбор дня консультации;
  - запись на свободный объект консультации.

### **2.2 Структура программного обеспечения**

Структура подсистемы в контексте автоматизированного проектирования (САПР) представляет собой организованное объединение компонентов, которые выполняют определенные функции и взаимодействуют друг с другом для достижения общих целей системы. Подсистема может быть самостоятельной частью более крупной системы или интегрированной в её архитектуру.

#### **2.2.1 Диаграммы классов**

Диаграммы классов – одна из основных диаграмм языка UML, которая показывают, какие существуют классы и связи между ними в логической структуре системы. На диаграмме можно также отобразить стереотип класса,

видимость всех атрибутов и операций, тип данных атрибутов и сигнатуру всех операций.

Диаграмма классов, разработанная в среде UML моделирования ArgoUML, представлена на рисунке 25.

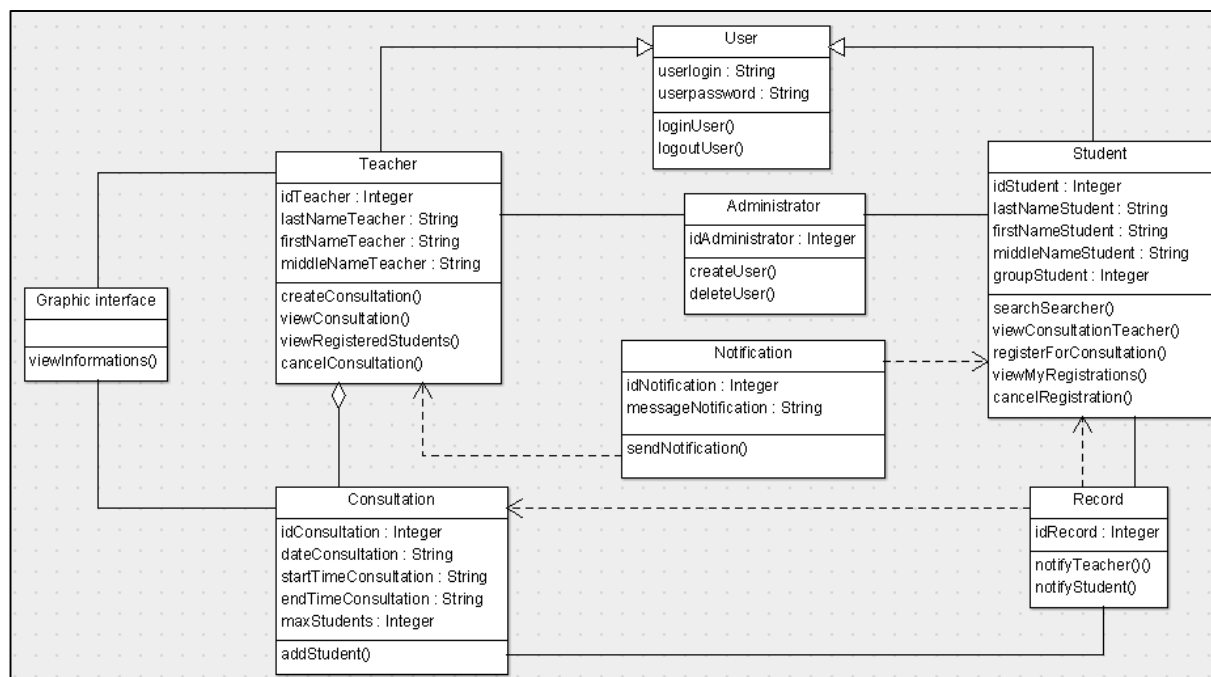


Рисунок 25 – Диаграмма классов

Эти классы формируют основу для системы управления учебным процессом, обеспечивая функциональность для администраторов, студентов и преподавателей.

За графический редактор отвечает класс GraphicInterface.

Класс Administrator, который управляет пользователями. Он содержит методы createUser и deleteUser, а также атрибуты, указывающие на объекты Teacher и Student.

Класс Consultation моделирует консультации, на которые могут записываться студенты. Он включает атрибуты dateConsultation, startTimeConsultation, endTimeConsultation, maxStudents и метод addStudent для добавления студентов на консультацию.

Класс Notification предназначен для отправки уведомлений. Он содержит метод sendNotification, а также атрибуты idNotification и messageNotification для хранения идентификатора и текста уведомления.

Класс Record представляет запись о консультации, которая может уведомлять студентов и преподавателей. Он содержит методы notifyStudents и notifyTeacher, а также атрибуты idRecord, consultation и student, которые связывают запись с консультацией и студентом.

Класс Student представляет студента и наследует от класса User. Он включает такие методы, как searchConsultations, viewConsultations, registerForConsultation и cancelRegistration, а также атрибуты idStudent и groupStudent для идентификации студента и его группы.

Класс Teacher представляет преподавателя и также наследует от класса User. Он включает методы create\_consultation и manage\_consultation, а также атрибуты teacher\_id и subject для идентификации преподавателя и предмета.

Класс User является базовым классом для пользователей системы и содержит методы loginUser и logoutUser.

В диаграмме классов представлены 4 типа связей. Ассоциация (association), наиболее общий и неопределённый вид отношений. Обычно в процессе детализации проекта, ассоциация превращается в какую-то специализированную связь. Наследование (inheritance) или обобщение (generalization), наиболее эффективный тип отношений, который используется, как для отражения общности, так и для отражения ассоциативности. Агрегация (aggregation), описание одного класса включает описание другого. Зависимостью (dependence) называют отношение использования, согласно которому изменение спецификации одного элемента может повлиять на другой элемент его использующий.

### **2.2.2 Диаграммы компонентов**

Диаграмма компонентов – диаграмма, на которой изображено множество компонентов и зависимости между ними; относится к статическому виду системы. Компонент – это физически заменяемая часть системы, совместимая с одним набором интерфейсов и обеспечивающая реализацию какого-либо другого интерфейса.

К имени компонента обычно добавляется расширение имен файлов. Отношения между компонентами – это зависимости, которые изображаются стрелкой, выходящей из зависимого модуля. Диаграммы компонентов могут также содержать пакеты или подсистемы для группирования элементов модели в более крупные блоки. Диаграмма компонентов дает представление о том, в каком порядке нужно компилировать компоненты, а также какие исполняемые компоненты будут созданы системой.

Диаграмма компонентов представлена на рисунке 26.

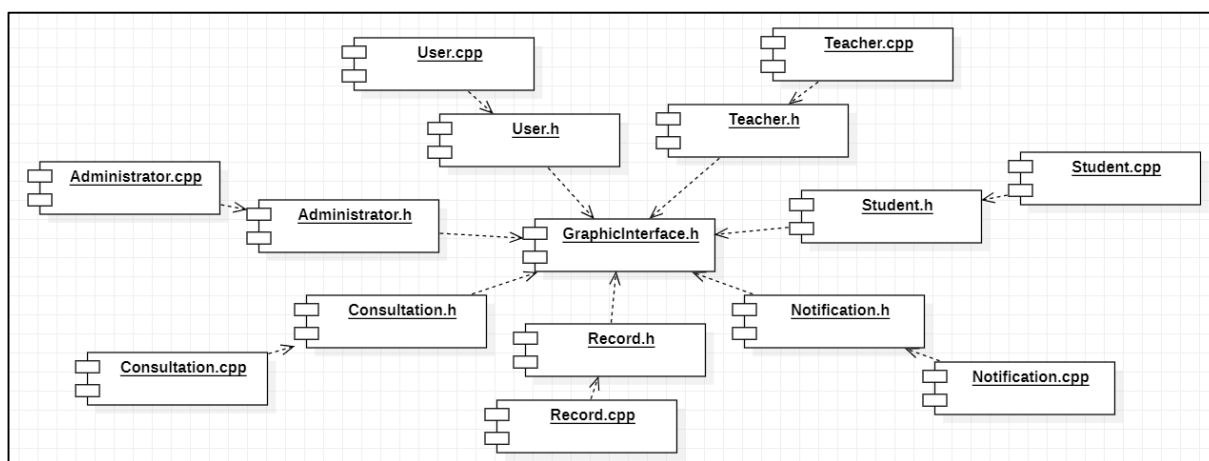


Рисунок 26 – Диаграмма компонентов

## 2.3 Схема алгоритма

Алгоритм представляет собой четкий и понятный набор инструкций, который может быть выполнен для достижения заданного результата. В программировании алгоритмы играют ключевую роль, так как они определяют логику работы программ и помогают структурировать их функциональность.

Кроме того, алгоритмы позволяют разработчикам лучше понимать и анализировать процессы, что упрощает выявление ошибок и оптимизацию кода. Они служат основой для разработки программного обеспечения, обеспечивая последовательность действий и предсказуемость результатов.

На рисунке 27 представлена главная схема алгоритма. Эта схема иллюстрирует основные этапы и ветвления в логике специализированного графического редактора.

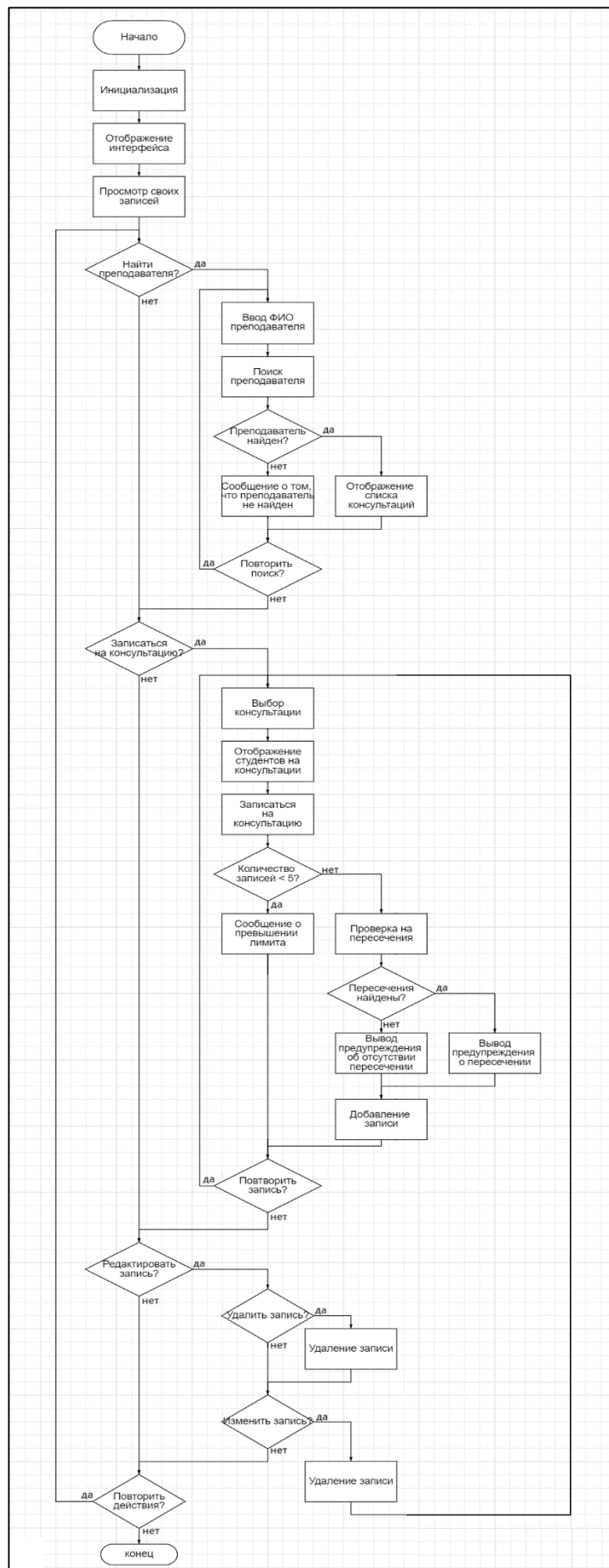


Рисунок 27 – Основная схема алгоритма

## 3 Разработка прототипа

Прототип реализован с использованием JavaFX и включает в себя основные функции, необходимые для взаимодействия пользователей с системой.

### 3.1 Отладка прототипа подсистемы

Отладка прототипа является важным этапом, позволяющим выявить и устранить ошибки, а также улучшить функциональность приложения. Код программы приведен в Приложении Б.

В процессе отладки были проведены следующие действия, представленные ниже.

Для того, чтобы записаться на консультацию, необходимо выбрать объект – преподаватель. Для этого в поле ввода нужно ввести его ФИО (рисунок 28).

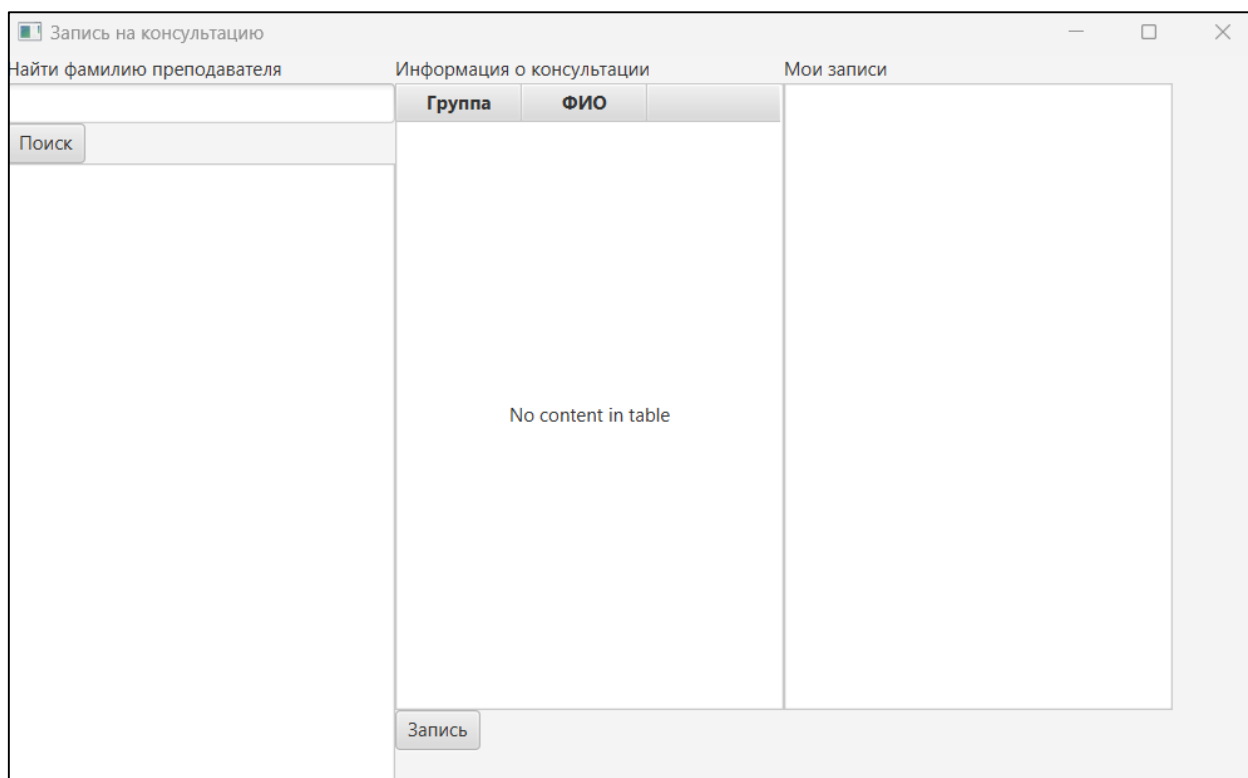


Рисунок 28 – Главное окно интерфейса

После того, как преподаватель был найден, выводится список его консультаций. Необходимо выбрать один день из консультаций (рисунок 29).

[illegible]

После того, как выбран день и время консультации выводится список записавшихся студентов. Для записи необходимо выбрать свободный объект – консультации (рисунок 30).

[illegible]

[illegible]

После того, как студент нажал на кнопку изменения записи необходимо так же выбрать свободный объект и записаться на консультацию. Но перед этим нужно выбрать другой день консультации. После записи появляется новая запись у студента и пропадает предупреждение о том, что консультации пересекаются (рисунок 32).

[illegible]

В другом случае, если других консультаций не найдено или по какой-то другой причине, можно удалить консультацию. После удаления пропадает запись в записях студента (рисунок 33).

[illegible]

Рисунок 33 – Удаление консультации

## Заключение

В ходе выполнения курсового проекта по разработке учебной системы автоматизированного проектирования (САПР) для учета и записи консультаций на кафедре была достигнута поставленная цель – создание интуитивно понятного и функционального графического интерфейса, который позволяет пользователям эффективно взаимодействовать с системой. Разработка специализированного графического редактора, реализованного на языке Java с использованием библиотеки JavaFX, продемонстрировала применение принципов объектно-ориентированного программирования и важность проектирования в создании пользовательских приложений.

В процессе работы над проектом было использовано множество современных инструментов и технологий, таких как StarUML и ArgoUML для построения диаграмм UML, а также Figma для проектирования интерфейса. Эти инструменты позволили не только визуализировать архитектуру системы, но и создать удобный и привлекательный интерфейс, что является ключевым аспектом успешного программного обеспечения.

Разработка системы учета консультаций улучшила организацию взаимодействия между студентами и преподавателями и повысила эффективность работы кафедры. В результате была создана система, которая обеспечивает легкость записи на консультации, а также предоставляет пользователям возможность управлять записями с минимальными усилиями.

Таким образом, проектирование систем автоматизированного проектирования не только способствует созданию качественного программного обеспечения, но и обеспечивает его соответствие потребностям пользователей, что является залогом успешной реализации любого проекта. В данной работе будет рассмотрен процесс разработки программы, ее функциональные возможности и результаты тестирования, что позволит глубже понять важность проектирования в контексте современных технологий.

## **Список используемой литературы**

1. Сегелен Э. Введение в базы данных. - М.: Вильямс, 2014.
2. Мак-Таггарт Р., Кримби Дж. MySQL: руководство по использованию. - СПб.: Диалектика, 2012.
3. Дубинин И.В. PHP. Создание динамических веб-сайтов. - М.: Эксмо, 2016.
4. Николас Закас. К совершенствованию современных веб-сайтов. - М.: ДМК Пресс, 2015.
5. Welling L., Thomson L. PHP и MySQL для начинающих. - СПб.: Питер, 2011.
6. Аллен Д. Основы разработки веб-приложений. - М.: ДМК Пресс, 2014.
7. Доулмэн З., Себастьян Б. PHP. Объекты, шаблоны и методики программирования. - СПб.: Питер, 2010.
8. Баклажан С., Баклажан К. Программирование на PHP. - М.: БХВ-Петербург, 2015.
9. Ланге Д., Миль Ж. PHP 7 в подлиннике. - М.: Вильямс, 2016.
10. Мак-Таггарт Р., Ашю Н., Урумов Н. MySQL Cookbook. - М.: ДМК Пресс, 2016.

# Приложение А

## Генерация SQL кода

### Administrator.h

```
#ifndef Administrator_h
#define Administrator_h

#include "User.h"

class Teacher;
class Student;

class Administrator : public User, public User, public User, public User {

public:

    virtual void createUser();

    virtual void deleteUser();

public:
    Integer idAdministrator;

public:

    /**
     * @element-type User
     */
    User *myUser;

    /**
     * @element-type Teacher
     */
    Teacher *myTeacher;

    /**
     * @element-type Student
     */
    Student *myStudent;
};

#endif // Administrator_h
```

### Consultation.h

```
#ifndef Consultation_h
#define Consultation_h

class Record;
class Teacher;
class Student;
class GraphicInterface;

class Consultation {

public:

    virtual void addStudent();

public:
```

```

    Integer idConsultation;
    String dateConsultation;
    String startTimeConsultation;
    String endTimeConsultation;
    Integer maxStudents;

public:

    /**
     * @element-type Record
     */
    Record *myRecord;

    /**
     * @element-type Teacher
     */
    Teacher *myTeacher;

    /**
     * @element-type Record
     */
    Record *myRecord;

    /**
     * @element-type Teacher
     */
    Teacher *myTeacher;

    /**
     * @element-type Student
     */
    Student *myStudent;

    /**
     * @element-type Record
     */
    Record *myRecord;

    /**
     * @element-type Teacher
     */
    Teacher *myTeacher;

    /**
     * @element-type GraphicInterface
     */
    GraphicInterface *myGraphicInterface;
};

#endif // Consultation_h

```

## GraphicInterface.h

```

#ifndef GraphicInterface_h
#define GraphicInterface_h

class Consultation;
class Teacher;

class GraphicInterface {

public:

```

```

        virtual void viewInformations();

public:

    /**
     * @element-type Consultation
     */
    Consultation *myConsultation;

    /**
     * @element-type Teacher
     */
    Teacher *myTeacher;
};

#endif // GraphicInterface_h

```

## Notification.h

```

#ifndef Notification_h
#define Notification_h

class Notification {

public:

    virtual void sendNotification();

public:
    Integer idNotification;
    String messageNotification;
};

#endif // Notification_h

```

## Record.h

```

#ifndef Record_h
#define Record_h

class Student;
class Consultation;

class Record {

public:

    virtual void notifyTeacher()();

    virtual void notifyStudent();

public:
    Integer idRecord;

public:

    /**
     * @element-type Student
     */
    Student *myStudent;

```

```

    /**
     * @element-type Consultation
     */
    Consultation *myConsultation;

    /**
     * @element-type Consultation
     */
    Consultation *myConsultation;

    /**
     * @element-type Student
     */
    Student *myStudent;
};

#endif // Record_h

```

## Student.h

```

#ifndef Student_h
#define Student_h

#include "User.h"

class Record;
class Administrator;
class Consultation;

class Student : public User, virtual public User {
public:
    virtual void searchSearcher();
    virtual void viewConsultationTeacher();
    virtual void registerForConsultation();
    virtual void viewMyRegistrations();
    virtual void cancelRegistration();

public:
    Integer idStudent;
    String lastNameStudent;
    String firstNameStudent;
    String middleNameStudent;
    Integer groupStudent;

public:
    /**
     * @element-type Record
     */
    Record *myRecord;

    /**
     * @element-type Administrator
     */
    Administrator *myAdministrator;

```

```

    /**
     * @element-type Consultation
     */
    Consultation *myConsultation;

    /**
     * @element-type Record
     */
    Record *myRecord;
};

#endif // Student_h

```

## Teacher.h

```

#ifndef Teacher_h
#define Teacher_h

#include "Student.h"
#include "User.h"

class Consultation;
class Administrator;
class GraphicInterface;

class Teacher : public Student, public User, virtual public User {
public:

    virtual void createConsultation();

    virtual void viewConsultation();

    virtual void viewRegisteredStudents();

    virtual void cancelConsultation();

public:
    Integer idTeacher;
    String lastNameTeacher;
    String firstNameTeacher;
    String middleNameTeacher;

public:

    /**
     * @element-type Consultation
     */
    Consultation *myConsultation;

    /**
     * @element-type Administrator
     */
    Administrator *myAdministrator;

    /**
     * @element-type Consultation
     */
    Consultation *myConsultation;

    /**
     * @element-type Consultation
     */

```

```

    Consultation *myConsultation;

    /**
     * @element-type GraphicInterface
     */
    GraphicInterface *myGraphicInterface;
};

#endif // Teacher_h

```

## User.h

```

#ifndef User_h
#define User_h

class User {

public:

    virtual void loginUser();

    virtual void logoutUser();

public:
    String userlogin;
    String userpassword;
};

#endif // User_h

```

# Приложение Б

## Программные модули

### GraphicInterface.java

```
package com.example.algorithmfx;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.stage.Stage;
import javafx.beans.property.SimpleStringProperty;

import java.util.ArrayList;
import java.util.List;

public class HelloApplication extends Application {

    private class StudentRecord {
        String group;
        String fullName;

        StudentRecord(String group, String fullName) {
            this.group = group;
            this.fullName = fullName;
        }
    }

    private class Consultation {
        String date;
        int maxStudents;
        List<StudentRecord> studentRecords;

        Consultation(String date, int maxStudents) {
            this.date = date;
            this.maxStudents = maxStudents;
            this.studentRecords = new ArrayList<>();
        }
    }

    private class Teacher {
        String name;
        List<Consultation> consultations;

        Teacher(String name) {
            this.name = name;
            this.consultations = new ArrayList<>();
        }
    }

    private class MyRecord {
        String teacherName;
        String consultationDate;
        StudentRecord studentRecord;

        MyRecord(String teacherName, String consultationDate, StudentRecord
studentRecord) {
            this.teacherName = teacherName;
            this.consultationDate = consultationDate;
            this.studentRecord = studentRecord;
        }
    }
}
```

```

    }
}

private List<Teacher> teachers = new ArrayList<>();
private ListView<String> consultationList;
private TableView<StudentRecord> studentTable;
private ListView<MyRecord> myRecords;
private List<MyRecord> myRecordList = new ArrayList<>();
private Label intersectionMessage;

@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Запись на консультацию");

    loadSampleData();

    VBox searchBox = new VBox();
    Label searchLabel = new Label("Найти фамилию преподавателя");
    TextField searchField = new TextField();
    Button searchButton = new Button("Поиск");

    consultationList = new ListView<>();
    consultationList.setOnMouseClicked(event ->
loadStudentTable(consultationList.getSelectionModel().getSelectedItem()));

    searchButton.setOnAction(e -> searchTeacher(searchField.getText()));

    searchBox.getChildren().addAll(searchLabel, searchField,
searchButton, consultationList);

    VBox consultationInfoBox = new VBox();
    Label consultationInfoLabel = new Label("Информация о консультации");
    studentTable = new TableView<>();
    TableColumn<StudentRecord, String> groupColumn = new
TableColumn<>("Группа");
    groupColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().group));
    TableColumn<StudentRecord, String> nameColumn = new
TableColumn<>("ФИО");
    nameColumn.setCellValueFactory(cellData -> new
SimpleStringProperty(cellData.getValue().fullName));
    studentTable.getColumns().addAll(groupColumn, nameColumn);

    Button registerButton = new Button("Запись");
    registerButton.setOnAction(e -> registerStudent());

    consultationInfoBox.getChildren().addAll(consultationInfoLabel,
studentTable, registerButton);

    VBox myRecordsBox = new VBox();
    Label myRecordsLabel = new Label("Мои записи");
    myRecords = new ListView<>();
    myRecords.setCellFactory(param -> new ListCell<MyRecord>() {
        @Override
        protected void updateItem(MyRecord item, boolean empty) {
            super.updateItem(item, empty);
            if (empty || item == null) {
                setText(null);
                setGraphic(null);
            } else {
                VBox recordBox = new VBox();
                recordBox.getChildren().add(new Label(item.teacherName +
"\n" + item.consultationDate));
            }
        }
    });
}

```

```

        Button deleteButton = new Button("Удалить");
        deleteButton.setOnAction(e -> deleteRecord(item));

        Button changeButton = new Button("Изменить");
        changeButton.setOnAction(e -> changeRecord(item));

        recordBox.getChildren().addAll(deleteButton,
changeButton);
        setGraphic(recordBox);
    }
}

intersectionMessage = new Label();
myRecordsBox.getChildren().addAll(myRecordsLabel, myRecords,
intersectionMessage);

// Основной layout
HBox mainLayout = new HBox(searchBox, consultationInfoBox,
myRecordsBox);
Scene scene = new Scene(mainLayout, 800, 600);
primaryStage.setScene(scene);
primaryStage.show();
}

private void loadSampleData() {
    Teacher teacher1 = new Teacher("Глотова Татьяна Владимировна");
    teacher1.consultations.add(new Consultation("пт, 02.11.24 13:15-
15:20", 5));
    teacher1.consultations.add(new Consultation("сб, 03.11.24 11:40-
13:45", 5));
    teacher1.consultations.get(0).studentRecords.add(new
StudentRecord("21BA1", "Петрова Н. В.));
    teacher1.consultations.get(0).studentRecords.add(new
StudentRecord("21BBC1", "Морозов С. К.));
    teacher1.consultations.get(0).studentRecords.add(new
StudentRecord("23BA1", "Петров К. В.));
    teacher1.consultations.get(1).studentRecords.add(new
StudentRecord("22BA1", "Сергеева А. А.));
    teacher1.consultations.get(1).studentRecords.add(new
StudentRecord("23BBC1", "Иванова М. И.));

    Teacher teacher2 = new Teacher("Гудков Алексей Анатольевич");
    teacher2.consultations.add(new Consultation("пт, 02.11.24 13:15-
15:20", 5));
    teacher2.consultations.get(0).studentRecords.add(new
StudentRecord("22BA1", "Шамин В. Д.));

    teachers.add(teacher1);
    teachers.add(teacher2);
}

private Teacher selectedTeacher;

private void searchTeacher(String name) {
    consultationList.getItems().clear();
    selectedTeacher = null;
    for (Teacher teacher : teachers) {
        if (teacher.name.contains(name)) {
            selectedTeacher = teacher;
            for (Consultation consultation : teacher.consultations) {
                consultationList.getItems().add(consultation.date);
            }
        }
    }
}

```

```

    }
    }
}

private void loadStudentTable(String consultationDate) {
    studentTable.getItems().clear();
    if (selectedTeacher != null) {
        for (Consultation consultation : selectedTeacher.consultations) {
            if (consultation.date.equals(consultationDate)) {

studentTable.getItems().addAll(consultation.studentRecords);

            }
        }
    }
}

private void registerStudent() {
    if (studentTable.getItems().size() < 5) {
        String group = "21BBC1";
        String fullName = "Нагорная Д. А.";
        StudentRecord newStudent = new StudentRecord(group, fullName);

        String selectedConsultationDate =
consultationList.getSelectionModel().getSelectedItem();
        String selectedTeacherName = selectedTeacher != null ?
selectedTeacher.name : null;

        boolean hasIntersection = false;
        for (MyRecord record : myRecordList) {
            if (record.consultationDate.equals(selectedConsultationDate)
&&
record.studentRecord.fullName.equals(newStudent.fullName)) {
                hasIntersection = true;
                break;
            }
        }

        if (hasIntersection) {
            intersectionMessage.setText("Найдены пересечения!");
            intersectionMessage.setStyle("-fx-text-fill: red;");
        } else {
            intersectionMessage.setText("Пересечения не найдены");
            intersectionMessage.setStyle("-fx-text-fill: green;");
        }

        if (selectedTeacher != null) {
            for (Consultation consultation :
selectedTeacher.consultations) {
                if (consultation.date.equals(selectedConsultationDate)) {
                    consultation.studentRecords.add(newStudent); //
Добавляем студента в консультацию
                    break;
                }
            }
        }

        if (selectedTeacherName != null) {
            MyRecord myRecord = new MyRecord(selectedTeacherName,
selectedConsultationDate, newStudent);
            myRecordList.add(myRecord);
            myRecords.getItems().clear();

```

```

        myRecords.getItems().addAll(myRecordList);
    }
    } else {
        intersectionMessage.setText("Максимальное количество записей достигнуто!");
        intersectionMessage.setStyle("-fx-text-fill: red;");
    }
}

private void deleteRecord(MyRecord record) {
    for (Teacher teacher : teachers) {
        for (Consultation consultation : teacher.consultations) {
            if (consultation.date.equals(record.consultationDate)) {
                consultation.studentRecords.removeIf(student ->
student.fullName.equals(record.studentRecord.fullName) &&
student.group.equals(record.studentRecord.group));
                break;
            }
        }
    }

    myRecordList.remove(record);
    myRecords.getItems().clear();
    myRecords.getItems().addAll(myRecordList);

    updateIntersectionMessage();
}

private void updateIntersectionMessage() {
    boolean hasIntersection = false;

    List<String> consultationDates = new ArrayList<>();
    for (MyRecord myRecord : myRecordList) {
        if (!consultationDates.contains(myRecord.consultationDate)) {
            consultationDates.add(myRecord.consultationDate);
        }
    }

    for (String consultationDate : consultationDates) {
        List<String> registeredStudents = new ArrayList<>();
        for (MyRecord myRecord : myRecordList) {
            if (myRecord.consultationDate.equals(consultationDate)) {
                registeredStudents.add(myRecord.studentRecord.fullName);
            }
        }

        if (registeredStudents.size() < myRecordList.size()) {
            hasIntersection = true;
            break;
        }
    }

    if (hasIntersection) {
        intersectionMessage.setText("Найдены пересечения!");
        intersectionMessage.setStyle("-fx-text-fill: red;");
    } else {
        intersectionMessage.setText("Пересечения не найдены");
        intersectionMessage.setStyle("-fx-text-fill: green;");
    }
}

private void changeRecord(MyRecord record) {

```

```
        deleteRecord(record);  
        registerStudent();  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```