МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ М.В. ЛОМОНОСОВА

Филиал МГУ имени М.В. Ломоносова в г. Душанбе

Кафедра математики и естественных наук

Отчет по учебной практике

Выполнил: студент 2-го курса

направления подготовки 01.03.04

Прикладная математика и информатика

Хайбулоев Д.А.

Научный руководитель: д.соц.н.,

профессор кафедры социологии

управления МГУ имени

М.В. Ломоносова Рыбакова М.В.

Душанбе - 2024

**1. Введение**

Учебная практика является важным этапом обучения студентов направления "Прикладная математика и информатика". Основной целью данной практики является закрепление теоретических знаний, полученных в процессе обучения, и приобретение практических навыков в области разработки программного обеспечения. Практика направлена на разработку программного обеспечения общего назначения, предназначенного для работы с расписанием университетов.

Создание программного обеспечения для управления университетским расписанием является актуальной задачей, так как современное образовательное учреждение требует автоматизации процессов планирования занятий, управления информацией о преподавателях, студентах, предметах и учебных аудиториях.

Разработанное программное обеспечение позволяет эффективно управлять различными аспектами учебного процесса. Оно включает в себя функциональные модули для работы с расписанием, распределения учебных дисциплин, а также управления информацией о преподавателях, студентах, группах и аудиториях. Система базируется на реляционной базе данных MS SQL, которая обеспечивает надежное хранение и обработку данных.

Целью данной работы является разработка программного обеспечения, которое автоматизирует процессы создания и управления расписанием в университете. В ходе выполнения учебной практики были поставлены следующие задачи:

* Проектирование структуры базы данных для хранения информации о преподавателях, студентах, учебных группах, аудиториях, учебных предметах и расписаниях.
* Разработка модулей для управления справочной информацией (преподаватели, предметы, группы, аудитории).
* Создание функциональности для составления расписания и его хранения в базе данных.
* Разработка интерфейсов для отображения расписания и поиска информации по нему.

Данный отчет описывает процесс выполнения учебной практики, включает описание разработанных модулей, структуры базы данных, а также основные моменты реализации кода. Отчет отражает все этапы разработки программного обеспечения, от анализа требований до реализации функциональных модулей, и демонстрирует достигнутые результаты.

**2. Анализ требований**

Анализ требований представляет собой важный этап разработки программного обеспечения, в ходе которого определяются функциональные и нефункциональные требования к системе. В данном разделе рассматриваются ключевые требования к разработанному программному обеспечению для работы с расписанием университетов.

**2.1 Функциональные требования**

Функциональные требования описывают, какие задачи должна выполнять система и какие функции она должна предоставлять пользователю. Для нашего программного обеспечения были определены следующие ключевые функциональные требования:

1. **Модуль "Справочники"**
   * **Преподаватели**:
     + **Добавление и удаление преподавателей**: Пользователь должен иметь возможность добавлять нового преподавателя в базу данных и удалять существующего.
     + **Просмотр и печать расписания**: Пользователь может просмотреть расписание преподавателя на следующую неделю и распечатать его при необходимости.
     + **Просмотр информации**: Возможность просмотра информации о преподавателе, включая ФИО, должность, контактные данные и другие важные сведения.
   * **Предметы**:
     + **Добавление и удаление предметов**: Позволяет добавлять новые предметы в базу данных и удалять устаревшие или неактуальные.
   * **Группы**:
     + **Управление группами**: Возможность добавления новых учебных групп, удаления существующих, а также просмотра информации о группах.
     + **Управление студентами**: Включает добавление и удаление студентов из групп, просмотр их личной информации и данных о посещаемости.
   * **Кабинеты**:
     + **Просмотр информации об аудиториях**: Пользователь может просмотреть информацию о доступных кабинетах, их расположении и вместимости.
     + **Добавление и удаление кабинетов**: Возможность добавления новых аудиторий в систему и удаления ненужных.
   * **Дни рождения (ДР)**:
     + **Просмотр дней рождения студентов**: Отображение дней рождения студентов на текущую неделю, что позволяет своевременно поздравить студентов и планировать мероприятия.
   * **Уведомления**:
     + **Установка уведомлений**: Пользователь может устанавливать уведомления с указанием даты и времени, которые будут отображаться в системе для напоминания о важных событиях.
2. **Модуль "Учебный план"**
   * **Распределение предметов по часам и семестрам**: Система должна обеспечивать распределение учебных дисциплин по часам и семестрам в соответствии с учебным планом университета.
   * **Назначение преподавателей**: Возможность распределения предметов между преподавателями с учетом их расписания и квалификации.
3. **Модуль "Составление расписания"**
   * **Создание расписания**: Позволяет пользователю создавать и редактировать расписание занятий, учитывая доступность преподавателей, аудиторий и учебных групп.
   * **Сохранение расписания**: Расписание должно сохраняться в базе данных и быть доступным для последующего просмотра и редактирования.
4. **Модуль "Показ расписания"**
   * **Отображение расписания**: Интерфейс для удобного просмотра расписания по дням, неделям и месяцам, с возможностью фильтрации и поиска по различным параметрам.
5. **Модуль "Поиск по расписанию"**
   * **Поиск информации**: Функциональность поиска по расписанию, которая позволяет находить занятия, преподавателей и аудитории на основе введенных параметров.

**2.2 Нефункциональные требования**

Нефункциональные требования определяют характеристики системы, которые не связаны напрямую с её функциональностью, но имеют критическое значение для её успешной работы.

1. **Производительность**: Быстрая обработка запросов при большом объёме данных, за счет правильной структуризации таблиц и БД в общем.
2. **Надежность**: Отказоустойчивость и сохранность данных при сбоях.
3. **Удобство использования:** Интуитивно понятный интерфейс для минимизации времени обучения.
4. **Безопасность:** Защита данных и управление правами доступа.
5. **Масштабируемость:** Возможность расширения функционала без значительных изменений архитектуры.

В результате выполнения анализа требований были определены ключевые функции и характеристики, необходимые для успешной реализации проекта. Это позволило создать четкое представление о том, что должно быть включено в программное обеспечение, и задать направление для последующих этапов разработки.

**3. Проектирование системы**

В этом разделе описана структура базы данных, которая используется в разработанном программном обеспечении для работы с расписанием университетов. Программное обеспечение основано на реляционной базе данных MS SQL и включает в себя несколько ключевых таблиц, каждая из которых выполняет свою роль в управлении учебной информацией.

**3.1 Описание базы данных**

**Таблица Diary**

Таблица Diary содержит информацию об уведомлениях. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | Int | Идентификатор уведомления |
| DataTime | Datetime | Дата и время уведомления |
| Name | Nvarchar | Название уведомления |
| Evant | Nvarchar | Событие уведомления |

**Таблица Kabin**

Таблица Kabin содержит информацию о кабинетах. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | Идентификатор аудитории |
| room | nvarchar | Номер аудитории |
| tipe | nvarchar | Тип аудитории |

**Таблица Groupe**

Таблица Groupe содержит информацию о группах и их специальностях. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| Id | int | Идентификатор группы |
| Name | nvarchar | Название группы |
| spec | nvarchar | Специальность |
| cours | int | Курс |

**Таблица Kaphedra**

Таблица Kaphedra содержит информацию о кафедрах университета. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | Идентификатор кафедры |
| Name | nvarchar | Название кафедры |

**Таблица Dist**

Таблица Dist хранит информацию о распределении учебных дисциплин по часам и семестрам. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| Id | Int | Идентификатор дисциплины |
| Name\_Dist | Nvarchar | Название дисциплины |
| Semestr | Int | Семестр |
| Cours | Int | Курс |
| Speciality\_Id | Int | Идентификатор специальности |
| Prepod\_Id | Int | Идентификатор преподавателя |
| In\_Audience | Int | Аудиторные часы |
| Lecture\_Hours | Int | Часы лекций |
| Practice\_Hours | Int | Часы практических занятий |
| Lab\_Hours | Int | Лабораторные часы |
| Semenars | Int | Часы семинаров |
| Independent\_Hours | Int | Часы самостоятельной работы |
| Form\_Of\_Control | Nvarchar | Форма контроля |

**Таблица Dist\_With\_Prepod**

Таблица Dist\_With\_Prepod содержит распределение дисциплин между преподавателями. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | Int | Идентификатор записи |
| Name\_Dist | Nvarchar | Название дисциплины |
| Semestr | Int | Семестр |
| Cours | Int | Курс |
| Speciality\_Id | Int | Идентификатор специальности |
| Prepod\_Id | Int | Идентификатор преподавателя |
| In\_Audience | Int | Аудиторные часы |
| Lecture\_Hours | Int | Часы лекций |
| Practice\_Hours | Int | Часы практических занятий |
| Lab\_Hours | Int | Лабораторные часы |
| Semenars | Int | Часы семинаров |
| Independent\_Hours | Int | Часы самостоятельной работы |
| Form\_Of\_Control | Nvarchar | Форма контроля |
| Control\_Hours | Int | Часы контроля |

**Таблица Speciality**

Таблица Speciality содержит информацию о специальностях. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | Int | Идентификатор специальности |
| Name | Nvarchar | Название специальности |

**Таблица Prepods**

Таблица Prepods хранит информацию о преподавателях. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | Int | Идентификатор преподавателя |
| First\_name | Nvarchar | Фамилия преподавателя |
| Name | Nvarchar | Имя преподавателя |
| Last\_name | Nvarchar | Отчество преподавателя |
| Faculty | Nvarchar | Факультет |
| Kaphedra | Int | Кафедра |
| Position | Nvarchar | Должность |
| Adress | Nvarchar | Адрес |
| Phone | Nvarchar | Телефон |
| Email | Nvarchar | Электронная почта |

**Таблица Subjects**

Таблица Subjects содержит информацию о предметах. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| Id | Int | Идентификатор предмета |
| Subjects | Nvarchar | Название предмета |

**Таблица Rasp**

Таблица Rasp содержит расписание. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | Int | Идентификатор записи |
| id\_kabin | Int | Идентификатор кабинета |
| data | Datetime | Дата |
| week | Int | Номер недели |
| year | Int | Год |
| week\_day | Nvarchar | День недели |
| id\_pred | Int | Идентификатор дисциплины |
| id\_prep | Int | Идентификатор преподавателя |
| id\_gr | Int | Идентификатор группы |
| para | Int | Пара |
| tipe | Nvarchar | Тип занятия |

**Таблица Students**

Таблица Students хранит информацию о студентах. Структура таблицы:

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | Int | Идентификатор студента |
| First\_name | Nvarchar | Фамилия студента |
| Name | Nvarchar | Имя студента |
| Last\_name | Nvarchar | Отчество студента |
| Cours | Int | Курс |
| Speciality | Nvarchar | Специальность |
| Adres | Nvarchar | Адрес |
| Telephone | Nvarchar | Телефон |
| Email | Nvarchar | Электронная почта |
| Birthday | Datetime | Дата рождения |

**4. Реализация функционала**

В этом разделе описывается реализация основных функций системы управления расписанием и учебными данными в университете. Программное обеспечение разработано на основе архитектуры клиент-сервер с использованием MS SQL для хранения данных и C++ Builder для реализации пользовательского интерфейса и логики взаимодействия с базой данных.

Система состоит из нескольких модулей, каждый из которых отвечает за выполнение специфических задач, необходимых для эффективного управления учебными процессами:

В данном разделе описывается реализация основных функций системы для управления расписанием, распределением учебной нагрузки и поддержанием актуальных данных о преподавателях, студентах и аудиториях. Программное обеспечение построено на архитектуре клиент-сервер с использованием MS SQL и C++ Builder.

Основные модули системы включают:

1. **Управление расписанием**: Обеспечивает добавление, изменение и удаление записей в расписании, а также поддержку его актуальности в случае изменений в учебном процессе.
2. **Управление дисциплинами и часами**: Позволяет администрировать учебные дисциплины и распределять учебные часы по семестрам и курсам, обеспечивая планирование учебного процесса.
3. **Управление преподавателями и студентами**: Модуль объединяет функционал для управления данными о преподавателях и студентах: добавление, обновление и удаление записей, поддержание актуальной информации для учебного процесса.
4. **Показ дней рождения студентов**: Функция для отображения информации о предстоящих днях рождения студентов в пределах недели, что помогает в организации поздравлений и подготовки учебного процесса.
5. **Управление группами и специальностями**: Поддерживает данные о студенческих группах и их специальностях, позволяя добавлять, редактировать и удалять записи для точного учета студентов.
6. **Управление уведомлениями**: Позволяет создавать и управлять уведомлениями о различных событиях, что способствует лучшей организации времени и повышению эффективности работы.

Каждый модуль системы реализован с использованием методов, взаимодействующих с базой данных через SQL-запросы для выполнения операций добавления, изменения или удаления данных. Код разработан с учетом принципов модульности и переиспользуемости, что обеспечивает легкость поддержки и расширения функционала системы.

В следующих разделах будут приведены примеры реализации функционала, а также основные SQL-запросы и фрагменты кода, используемые для взаимодействия с базой данных.

**4.1 Управление расписанием**

Функция управления расписанием позволяет добавлять, изменять и удалять записи расписания в базе данных. Эта функция обеспечивает возможность поддержания актуального расписания занятий, позволяя пользователю вносить изменения по мере необходимости.

*//Получения нужного формата даты для бд и семестр*

*String top, dst2, year;*

*top = DateTimePicker2->Date.FormatString("yyyy"); // Получаем год*

*int yo = StrToInt(top);*

*int semest = StrToInt(ComboBox6->Text);*

*if (semest % 2 != 0) {*

*top = IntToStr(yo) + "-" + IntToStr(yo + 1);*

*} else {*

*top = IntToStr(yo - 1) + "-" + IntToStr(yo);*

*}*

*dst2 = DateTimePicker2->Date.FormatString("dd.MM.yyyy"); // Форматируем дату*

*//Получаем фамилию и инициалы преподавателя для поиска его в БД*

*String fullName = ComboBox2->Text;*

*TStringList\* nameParts = new TStringList();*

*nameParts->Delimiter = ' ';*

*nameParts->DelimitedText = fullName;*

*String lastName = nameParts->Strings[0];*

*String firstInitial = nameParts->Strings[1].SubString(1, 1);*

*String middleInitial = nameParts->Strings[2].SubString(1, 1);*

*delete nameParts;*

*//Формируем запрос на проверку есть ли данная запись в БД*

*ADOQuery1->Close();*

*ADOQuery1->SQL->Clear();*

*ADOQuery1->SQL->Add("USE Practic\_work");*

*ADOQuery1->SQL->Add("SELECT Rasp.\*, Kabin.room, Prepods.\*, Subjects.\*, Dist\_With\_Prepod.semestr, Groupe.name, Groupe.cours");*

*ADOQuery1->SQL->Add("FROM Rasp");*

*ADOQuery1->SQL->Add("INNER JOIN Kabin ON Rasp.id\_kabin = Kabin.id");*

*ADOQuery1->SQL->Add("INNER JOIN Subjects ON Rasp.id\_pred = Subjects.id");*

*ADOQuery1->SQL->Add("INNER JOIN Prepods ON Rasp.id\_prep = Prepods.id");*

*ADOQuery1->SQL->Add("INNER JOIN Groupe ON Rasp.id\_gr = Groupe.id");*

*ADOQuery1->SQL->Add("INNER JOIN Dist\_With\_Prepod ON Rasp.id\_pred = Dist\_With\_Prepod.Name\_Dist");*

*ADOQuery1->SQL->Add("WHERE para = :year AND week\_day = :week\_day AND week = :week AND data = :data AND year = :year\_string AND room = :room");*

*ADOQuery1->Parameters->ParamByName("year")->Value = year;*

*ADOQuery1->Parameters->ParamByName("week\_day")->Value = Edit6->Text;*

*ADOQuery1->Parameters->ParamByName("week")->Value = Edit7->Text;*

*ADOQuery1->Parameters->ParamByName("data")->Value = dst2;*

*ADOQuery1->Parameters->ParamByName("year\_string")->Value = top;*

*ADOQuery1->Parameters->ParamByName("room")->Value = ComboBox3->Text;*

*ADOQuery1->Open();*

*if (!ADOQuery1->IsEmpty()) {*

*Memo1->Clear();*

*Memo1->Lines->Add("!!!Внимание!!!");*

*Memo1->Lines->Add("В расписании уже есть эта запись, перепроверьте данные и в частности дата)");*

*} else {*

*//Проверка Преподавателя и его пар, не пересекается ли что-то в БД*

*ExecuteSQLQueryForOverlap(year, lastName, firstInitial, middleInitial, dst2, top, ComboBox3->Text, ComboBox1->Text, ComboBox4->Text, ComboBox5->Text, Edit6->Text, Edit7->Text, Edit15->Text); }*

*void ExecuteSQLQueryForOverlap(String year, String lastName, String firstInitial, String middleInitial, String dst2, String top, String room, String subject, String groupName, String course, String weekDay, String week, String type) {*

*ADOQuery1->Close();*

*ADOQuery1->SQL->Clear();*

*ADOQuery1->SQL->Add("USE Practic\_work");*

*ADOQuery1->SQL->Add("SELECT Rasp.\*, Kabin.room, Prepods.\*, Subjects.\*, Dist\_With\_Prepod.semestr, Groupe.name, Groupe.cours");*

*ADOQuery1->SQL->Add("FROM Rasp");*

*ADOQuery1->SQL->Add("INNER JOIN Kabin ON Rasp.id\_kabin = Kabin.id");*

*ADOQuery1->SQL->Add("INNER JOIN Subjects ON Rasp.id\_pred = Subjects.id");*

*ADOQuery1->SQL->Add("INNER JOIN Prepods ON Rasp.id\_prep = Prepods.id");*

*ADOQuery1->SQL->Add("INNER JOIN Groupe ON Rasp.id\_gr = Groupe.id");*

*ADOQuery1->SQL->Add("INNER JOIN Dist\_With\_Prepod ON Rasp.id\_pred = Dist\_With\_Prepod.Name\_Dist");*

*ADOQuery1->SQL->Add("WHERE para = :year AND week\_day = :week\_day AND week = :week AND data = :data AND year = :year\_string");*

*ADOQuery1->Parameters->ParamByName("year")->Value = year;*

*ADOQuery1->Parameters->ParamByName("week\_day")->Value = weekDay;*

*ADOQuery1->Parameters->ParamByName("week")->Value = week;*

*ADOQuery1->Parameters->ParamByName("data")->Value = dst2;*

*ADOQuery1->Parameters->ParamByName("year\_string")->Value = top;*

*ADOQuery1->Open();*

*if (!ADOQuery1->IsEmpty()) {*

*Memo1->Lines->Add("!!!Внимание!!!");*

*Memo1->Lines->Add("В расписании этой парой у этого преподавателя уже есть занятия, перепроверьте данные");*

*} else {*

*// Продолжить с добавлением новой записи*

*AddNewRaspRecord(year, dst2, week, top, weekDay, subject, lastName, firstInitial, middleInitial, room, groupName, course, type); }}*

*//Функция для добавления записи в БД*

*void AddNewRaspRecord(String year, String dst2, String week, String top, String weekDay, String subject, String lastName, String firstInitial, String middleInitial, String room, String groupName, String course, String type) {*

*// Получаем IDs для вставки*

*String id\_prepod = GetIdByField("Prepods", "First\_Name", lastName, firstInitial, middleInitial);*

*String id\_predmet = GetIdByField("Subjects", "subjects", subject);*

*String id\_gr = GetIdByField("Groupe", "name", groupName, "cours", course);*

*String id\_kabinet = GetIdByField("kabin", "room", room);*

*// Проверка на наличие конфликтов в расписании*

*ADOQuery1->Close();*

*ADOQuery1->SQL->Clear();*

*ADOQuery1->SQL->Add("USE Practic\_work");*

*ADOQuery1->SQL->Add("SELECT \* FROM Rasp WHERE week = :week AND week\_day = :week\_day AND id\_gr = :id\_gr AND para = :para");*

*ADOQuery1->Parameters->ParamByName("week")->Value = week;*

*ADOQuery1->Parameters->ParamByName("week\_day")->Value = weekDay;*

*ADOQuery1->Parameters->ParamByName("id\_gr")->Value = id\_gr;*

*ADOQuery1->Parameters->ParamByName("para")->Value = year;*

*ADOQuery1->Open();*

*if (!ADOQuery1->IsEmpty()) {*

*Memo1->Lines->Add("!!!Внимание!!!");*

*Memo1->Lines->Add("В введеной в вами неделе уже есть такой день недели, перепроверьте номер недели)");*

*} else {*

*ADOQuery1->Close();*

*ADOQuery1->SQL->Clear();*

*ADOQuery1->SQL->Add("USE Practic\_work");*

*ADOQuery1->SQL->Add("INSERT INTO Rasp (id\_kabin, data, week, year, week\_day, id\_pred, id\_prep, id\_gr, para, tipe)");*

*ADOQuery1->SQL->Add("VALUES(:id\_kabin, :data, :week, :year, :week\_day, :id\_pred, :id\_prep, :id\_gr, :para, :type)");*

*ADOQuery1->Parameters->ParamByName("id\_kabin")->Value = id\_kabinet;*

*ADOQuery1->Parameters->ParamByName("data")->Value = dst2;*

*ADOQuery1->Parameters->ParamByName("week")->Value = week;*

*ADOQuery1->Parameters->ParamByName("year")->Value = top;*

*ADOQuery1->Parameters->ParamByName("week\_day")->Value = weekDay;*

*ADOQuery1->Parameters->ParamByName("id\_pred")->Value = id\_predmet;*

*ADOQuery1->Parameters->ParamByName("id\_prep")->Value = id\_prepod;*

*ADOQuery1->Parameters->ParamByName("id\_gr")->Value = id\_gr;*

*ADOQuery1->Parameters->ParamByName("para")->Value = year;*

*ADOQuery1->Parameters->ParamByName("type")->Value = type;*

*ADOQuery1->ExecSQL();*

*Memo1->Clear();*

*Memo1->Lines->Add("Данные успешно добавлены"); }}*

*//Функция для получения ID*

*String GetIdByField(String tableName, String fieldName, String value, String extraFieldName = "", String extraValue = "") {*

*ADOQuery2->Close();*

*ADOQuery2->SQL->Clear();*

*ADOQuery2->SQL->Add("SELECT id FROM " + tableName + " WHERE " + fieldName + " = :value");*

*if (!extraFieldName.IsEmpty()) {*

*ADOQuery2->SQL->Add(" AND " + extraFieldName + " = :extra\_value");*

*ADOQuery2->Parameters->ParamByName("extra\_value")->Value = extraValue;*

*}*

*ADOQuery2->Parameters->ParamByName("value")->Value = value;*

*ADOQuery2->Open();*

*if (!ADOQuery2->IsEmpty()) return ADOQuery2->FieldByName("id")->AsString;*

*return "";*

*}*

Этот код выполняет следующие операции:

1. Формирование даты и семестра: Определяет формат даты и семестра на основе выбранных значений.

2. Получение данных преподавателя: Разделяет ФИО преподавателя на фамилию и инициалы.

3. Проверка на дублирование: Проверяет, существует ли уже запись с заданными параметрами в расписании, чтобы избежать дублирования.

4. Проверка на пересечения: Проверяет, не пересекаются ли новые занятия с уже существующими по указанному преподавателю.

5. Добавление новой записи: Если пересечений не обнаружено, добавляет новую запись в таблицу расписания.

Код организует проверку и обновление расписания, взаимодействуя с базой данных для обеспечения корректности и актуальности данных.

**4.2 Управление дисциплинами и часами**

Функционал управления дисциплинами и распределением учебных часов позволяет добавлять новые дисциплины, удалять и изменять существующие, а также распределять учебные часы между дисциплинами в зависимости от учебных планов.

*int p = MessageDlg("Добавить данную Дисциплину?", mtConfirmation, TMsgDlgButtons() << mbOK << mbCancel, 0);*

*if (p == mrOk) {*

*int cours = (StrToInt(Semestr->Text) + 1) / 2;*

*auto getValueOrNull = [](TEdit\* edit) { return edit->Text.IsEmpty() ? "NULL" : edit->Text; };*

*String addQuery = String("use Practic\_work\n"*

*"insert into Dist(Name\_Dist, Semestr, Cours, Speciality\_Id, Prepod\_Id, In\_Audience, Lecture\_Hours, Practice\_Hours, Lab\_Hours, Semenars, Independent\_Hours, Form\_Of\_Control) "*

*"values((SELECT id FROM Subjects WHERE subjects = '") + Dissipl->Text + "')," + Semestr->Text + "," + cours +*

*", (SELECT id FROM Speciality WHERE Name = '" + Napr\_search->Text + "'), NULL," +*

*getValueOrNull(audit) + "," + getValueOrNull(lec) + "," + getValueOrNull(prac) + "," +*

*getValueOrNull(lab) + "," + getValueOrNull(semenar) + "," + getValueOrNull(sam) +*

*", (SELECT id FROM Form\_of\_Control where name = '" + contr->Text + "')");*

*ADOQuery1->Close();*

*ADOQuery1->SQL->Clear();*

*ADOQuery1->SQL->Add(addQuery);*

*try {*

*ADOQuery1->ExecSQL();*

*ShowMessage("Добавление прошло успешно");*

*} catch (Exception &e) {ShowMessage("Ошибка выполнения запроса: " + e.Message);}*

*ADOQuery1->SQL->Clear();*

*ADOQuery1->SQL->Add("SELECT \* FROM Dist "*

*"JOIN Form\_Of\_Control ON Dist.Form\_Of\_Control = Form\_Of\_Control.id "*

*"JOIN Speciality ON Dist.Speciality\_Id = Speciality.id "*

*"JOIN Subjects ON Dist.Name\_Dist = Subjects.id");*

*try {*

*ADOQuery1->Open();*

*} catch (Exception &e) {ShowMessage("Ошибка выполнения запроса: " + e.Message) }}.*

Код для добавления дисциплины и распределения часов практически идентичны в силу одинаковой структуры их таблиц, а именно для дисциплин используется таблица Dist, а для распределения часов – Dist\_With\_Prepod, различие только в этом.

**4.3 Управление преподавателями и студентами:**

Управление данными о преподавателях включает функции добавления, редактирования и удаления информации о преподавателях. Это позволяет поддерживать актуальную базу данных сотрудников.

Пример кода добавления преподавателя в таблицу Prepods.

*// Переменная для хранения ID кафедры*

*int KaphedraID;*

*// Получение ID кафедры*

*ADOQueryPrepods->Close();*

*ADOQueryPrepods->SQL->Text = "SELECT id FROM Kaphedra WHERE Name = :KaphedraName";*

*ADOQueryPrepods->Parameters->ParamByName("KaphedraName")->Value = Edit5->Text;*

*try {*

*ADOQueryPrepods->Open();*

*if (!ADOQueryPrepods->Eof)*

*KaphedraID = ADOQueryPrepods->FieldByName("id")->AsInteger;*

*else { ShowMessage("Кафедра не найдена."); return; }*

*ADOQueryPrepods->Close();*

*} catch (const Exception &e) { ShowMessage("Ошибка: " + e.Message); return; }*

*// Добавление преподавателя*

*ADOQueryPrepods->SQL->Text =*

*"INSERT INTO Prepods (First\_name, Name, Last\_name, Faculty, Kaphedra, Position, Adress, Phone, Email) "*

*"VALUES (:First\_name, :Name, :Last\_name, :Faculty, :Kaphedra, :Position, :Adress, :Phone, :Email)";*

*ADOQueryPrepods->Parameters->ParamByName("First\_name")->Value = Edit1->Text;*

*ADOQueryPrepods->Parameters->ParamByName("Name")->Value = Edit2->Text;*

*ADOQueryPrepods->Parameters->ParamByName("Last\_name")->Value = Edit3->Text;*

*ADOQueryPrepods->Parameters->ParamByName("Faculty")->Value = Edit4->Text;*

*ADOQueryPrepods->Parameters->ParamByName("Kaphedra")->Value = KaphedraID;*

*ADOQueryPrepods->Parameters->ParamByName("Position")->Value = Edit6->Text;*

*ADOQueryPrepods->Parameters->ParamByName("Adress")->Value = Edit7->Text;*

*ADOQueryPrepods->Parameters->ParamByName("Phone")->Value = Edit8->Text;*

*ADOQueryPrepods->Parameters->ParamByName("Email")->Value = Edit9->Text;*

*try {*

*ADOQueryPrepods->ExecSQL();*

*ShowMessage("Преподаватель добавлен.");*

*} catch (const Exception &e) { ShowMessage("Ошибка: " + e.Message); }*

*// Обновление данных*

*ADOQueryPrepods->Close();*

*ADOQueryPrepods->SQL->Text =*

*"SELECT \* FROM Prepods "*

*"JOIN Kaphedra ON Prepods.Kaphedra = Kaphedra.id "*

*"ORDER BY First\_name";*

*try { ADOQueryPrepods->Open(); }*

*catch (const Exception &e) { ShowMessage("Ошибка: " + e.Message); }*

Код для добавления студентов:

int specialityId;

// Получение ID специальности

ADOQuerySpeciality->Close();

ADOQuerySpeciality->SQL->Text = "SELECT id FROM Speciality WHERE Name = :name";

ADOQuerySpeciality->Parameters->ParamByName("name")->Value = Edit1->Text;

ADOQuerySpeciality->Open();

if (ADOQuerySpeciality->IsEmpty()) {

ShowMessage("Специальность не найдена.");

ADOQuerySpeciality->Close();

return;

}

specialityId = ADOQuerySpeciality->FieldByName("id")->AsInteger;

ADOQuerySpeciality->Close();

// Форматирование даты рождения

String birthday = Edit10->Text.SubString(4,2) + "." + Edit10->Text.SubString(1,2) + "." + Edit10->Text.SubString(7,4);

try {

// Вставка новой записи студента

ADOQueryStudents->Close();

ADOQueryStudents->SQL->Text =

"INSERT INTO Students (First\_name, Name, Last\_name, Cours, Speciality, Adres, Telephone, Email, Birthday) "

"VALUES (:First\_name, :Name, :Last\_name, :Cours, :Speciality, :Adres, :Telephone, :Email, :Birthday)";

ADOQueryStudents->Parameters->ParamByName("First\_name")->Value = Edit4->Text;

ADOQueryStudents->Parameters->ParamByName("Name")->Value = Edit5->Text;

ADOQueryStudents->Parameters->ParamByName("Last\_name")->Value = Edit6->Text;

ADOQueryStudents->Parameters->ParamByName("Cours")->Value = Edit3->Text;

ADOQueryStudents->Parameters->ParamByName("Speciality")->Value = specialityId;

ADOQueryStudents->Parameters->ParamByName("Adres")->Value = Edit7->Text;

ADOQueryStudents->Parameters->ParamByName("Telephone")->Value = Edit8->Text;

ADOQueryStudents->Parameters->ParamByName("Email")->Value = Edit9->Text;

ADOQueryStudents->Parameters->ParamByName("Birthday")->Value = birthday;

ADOQueryStudents->ExecSQL();

// Обновление данных

ADOQueryStudents->Close();

ADOQueryStudents->SQL->Text =

"SELECT \* FROM Students "

"WHERE Speciality = :speciality AND Cours = :cours";

ADOQueryStudents->Parameters->ParamByName("speciality")->Value = specialityId;

ADOQueryStudents->Parameters->ParamByName("cours")->Value = Edit3->Text;

ADOQueryStudents->Open();

ShowMessage("Студент успешно добавлен.");

}

catch (const Exception &e) {

ShowMessage("Ошибка при добавлении студента: " + e.Message);}

Этот код идентичен вышенаписанному, единственное различие в структуре таблиц, а также приведение даты дня рождения студента к нужному формату, чтобы не возникали конфликты при добавлении записи в таблицу БД, а сам код приведения даты следующий:

*String dst2, dst3, year, birthday;*

*dst2 = Edit10->Text;*

*dst3 = dst2.SubString(4,2);*

*dst2.Delete(3,3);*

*dst2 = dst3 + "." + dst2;*

*dst2.Delete(11,9);*

*dst3=dst2;*

*dst3.Delete(3,8);*

*birthday = dst2;*

Почему это справедливо? Так как при давление в БД MS SQL SERVER для типа Date используется американский формат, а именно “mm.dd.yyyy”, а для стран СНГ привичен такой формат: “dd.mm.yyyy”, поэтому во избежания дискомфорта для пользователя используется данный код.

Что насчет удаления? Его можно реализовать двумя способами, искать нужного студента или преподавателя по данным, которые загружаются в соответсвующие поля после нажатия на него или же объявлять глобальную переменную, записывать туда идентификатор студента или преподавателя при нажатии на запись и потом удалять, уже используя этот идентификатор из глобальной переменной, примеры ниже:

*int specialityId;*

*ADOQuerySpeciality->Close();*

*ADOQuerySpeciality->SQL->Text = "SELECT id FROM Speciality WHERE Name = :name";*

*ADOQuerySpeciality->Parameters->ParamByName("name")->Value = Edit1->Text;*

*ADOQuerySpeciality->Open();*

*if (ADOQuerySpeciality->IsEmpty()) {*

*ShowMessage("Специальность не найдена.");*

*ADOQuerySpeciality->Close();*

*return;}*

*specialityId = ADOQuerySpeciality->FieldByName("id")->AsInteger;*

*ADOQuerySpeciality->Close();*

*ADOQuerySpeciality->SQL->Clear();*

*try {*

*ADOQueryStudents->Close();*

*ADOQueryStudents->SQL->Text =*

*"DELETE FROM Students WHERE First\_name = :First\_name AND Name = :Name AND Last\_name = :Last\_name "*

*"AND Cours = :Cours AND Speciality = :Speciality AND Adres = :Adres AND Telephone = :Telephone "*

*"AND Email = :Email AND Birthday = :Birthday";*

*ADOQueryStudents->Parameters->ParamByName("First\_name")->Value = Edit4->Text;*

*ADOQueryStudents->Parameters->ParamByName("Name")->Value = Edit5->Text;*

*ADOQueryStudents->Parameters->ParamByName("Last\_name")->Value = Edit6->Text;*

*ADOQueryStudents->Parameters->ParamByName("Cours")->Value = Edit3->Text;*

*ADOQueryStudents->Parameters->ParamByName("Speciality")->Value = specialityId;*

*ADOQueryStudents->Parameters->ParamByName("Adres")->Value = Edit7->Text;*

*ADOQueryStudents->Parameters->ParamByName("Telephone")->Value = Edit8->Text;*

*ADOQueryStudents->Parameters->ParamByName("Email")->Value = Edit9->Text;*

*ADOQueryStudents->Parameters->ParamByName("Birthday")->Value = Edit10->Text;*

*ADOQueryStudents->ExecSQL();*

*ADOQueryStudents->SQL->Text =*

*"SELECT \* FROM Students WHERE Speciality = (SELECT id FROM Speciality WHERE Name = :name) AND Cours = :cours";*

*ADOQueryStudents->Parameters->ParamByName("name")->Value = Edit1->Text;*

*ADOQueryStudents->Parameters->ParamByName("cours")->Value = Edit3->Text;*

*ADOQueryStudents->Open();*

*ShowMessage("Студент успешно удален.");*

*} catch (const Exception &e) {ShowMessage("Ошибка при удалении студента: " + e.Message);}*

А вот пример с использованием глобальной переменной:

*if (selectedTeacherId > 0) {*

*ADOQueryPrepods->Close();*

*ADOQueryPrepods->SQL->Text = "DELETE FROM Prepods WHERE id = :id";*

*ADOQueryPrepods->Parameters->ParamByName("id")->Value = selectedTeacherId;*

*ADOQueryPrepods->ExecSQL();*

*ADOQueryPrepods->SQL->Text = "SELECT \* FROM Prepods JOIN Kaphedra ON Prepods.Kaphedra = Kaphedra.id ORDER BY First\_name";*

*ADOQueryPrepods->Open();*

*ShowMessage("Преподаватель успешно удален.");*

*}else {ShowMessage("Пожалуйста, выберите преподавателя для удаления.");}*

Здесь *selectedTeacherId* как раз и является той самой глобальной переменной.

Каждый из этих подходов имеет свои плюсы и минусы, давайте рассмотрим их:

**Поиск по данным**

**Плюсы:**

* **Гибкость:** Подходит, если данные могут изменяться, и удаление зависит от их текущего состояния.
* **Меньше зависимости:** Не требуется хранить идентификатор в глобальной переменной, что делает код менее зависимым от состояния глобальных переменных.
* **Точность:** Можно точно определить запись по полям, которые могут включать дополнительные условия для поиска.

**Минусы:**

* **Производительность:** Поиск по данным может быть менее эффективным, особенно если данных много и поиск осуществляется по нескольким полям.
* **Ошибки данных:** Возможность ошибки при вводе данных или несовпадения полей может привести к некорректному удалению записей.

**Использование глобальной переменной для хранения идентификатора**

**Плюсы:**

* **Производительность:** Удаление по идентификатору обычно быстрее, так как идентификатор является уникальным и индексированным в базе данных.
* **Простота кода:** Упрощает код удаления, так как нет необходимости в дополнительных запросах для поиска записи.
* **Снижение риска ошибок:** Использование уникального идентификатора минимизирует риск ошибок при удалении.

**Минусы:**

* **Зависимость от состояния:** Глобальная переменная требует управления состоянием и может быть изменена в непредвиденный момент, что может привести к ошибкам.
* **Меньше гибкости:** При изменении структуры данных или требований может потребоваться изменение логики работы с идентификаторами.

Резюмирую, имеем: Оба подхода имеют свои преимущества и недостатки. Использование глобальной переменной для хранения идентификатора предпочтительнее, если:

* Идентификаторы легко доступны и надежно управляются.
* Проект требует высокой производительности и минимизации ошибок.

Поиск по данным более подходящ, если:

* Данные могут меняться или содержат дополнительные условия.
* Код требует гибкости и независимости от состояния глобальных переменных.

**4.4 Показ дней рождения студентов:**

Функционал управления днями рождениями включает возможность просмотра надвигающихся днях рождениях студентов на ближайшую неделю. Здесь главное не сам код в проект, а sql-запрос, который я представлю ниже, это самое основное в этом модуле.

*DECLARE @Today DATE = 'today\_date';*

*DECLARE @EndDate DATE = 'today\_date + 7';*

*SELECT first\_name, name, last\_name, birthday*

*FROM students*

*WHERE*

*(*

*(MONTH(birthday) = MONTH(@Today) AND DAY(birthday) >= DAY(@Today)) -- дни рождения в текущем месяце в период между @Today и концом месяца*

*OR*

*(MONTH(birthday) = MONTH(@EndDate) AND DAY(birthday) <= DAY(@EndDate)) -- дни рождения в следующем месяце в период между началом месяца и @EndDate*

*)*

*AND*

*(*

*(MONTH(birthday) > MONTH(@Today) OR (MONTH(birthday) = MONTH(@Today) AND DAY(birthday) >= DAY(@Today)))*

*AND*

*(MONTH(birthday) < MONTH(@EndDate) OR (MONTH(birthday) = MONTH(@EndDate) AND DAY(birthday) <= DAY(@EndDate)))*

*)*

*ORDER BY*

*CASE*

*WHEN MONTH(birthday) = MONTH(@Today) THEN DAY(birthday) - DAY(@Today)*

*ELSE DAY(birthday) + (DAY(EOMONTH(@Today)) - DAY(@Today) + 1)*

*END;*

В начале получаем даты поиска дней рождений студентов в пределе 7 дней, то есть одной недели. Дальше делаем запрос в таблицу Students по полям *first\_name, name, last\_name, birthday.* И создаем условие в нескольких частях. *Первая часть условия:* Это условие позволяет отфильтровать дни рождения, которые находятся в пределах следующей недели:

Если день рождения находится в том же месяце, что и @Today, тогда он должен быть после или на текущем дне.

Если день рождения находится в том же месяце, что и @EndDate, тогда он должен быть до или на дне @EndDate.

*Вторая часть условия:* Это условие делает дополнительную проверку, чтобы убедиться, что дни рождения находятся строго в диапазоне между @Today и @EndDate, независимо от того, находятся ли они в одном или в разных месяцах.

*Сортировка:* Сортировка выполняется по возрастанию:

Если день рождения в текущем месяце (@Today), то он отсортирован по разнице в днях от текущей даты.

Если день рождения в следующем месяце, то к дню месяца добавляется количество оставшихся дней в текущем месяце для правильной сортировки.

**4.5 Управление группами и специальностями**

Управление группами и специальностями студентов позволяет добавлять и удалять группы, а также обновлять информацию о студентах в зависимости от текущих данных.

**1.Добавление записи**

**Добавление новой группы:**

*INSERT INTO Groupe (Name, spec, cours)*

*VALUES (:Name, :spec, :cours);*

**Добавление новой специальности:**

*INSERT INTO Speciality (Name)*

*VALUES (:Name);*

**2.Редактирование записи**

**Редактирование группы:**

*UPDATE Groupe*

*SET Name = :Name, spec = :spec, cours = :cours*

*WHERE Id = :Id;*

**Редактирование специальности:**

*UPDATE Speciality*

*SET Name = :Name*

*WHERE id = :id;*

**3.Удаление записи**

**Удаление группы:**

*DELETE FROM Groupe*

*WHERE Id = :Id;*

**Удаление специальности:**

*DELETE FROM Speciality*

*WHERE id = :id;*

**4.Просмотр групп по специальности:**

*SELECT \* FROM Groupe*

*WHERE spec = :spec;*

**5.Просмотр специальностей по группе:**

*SELECT s.\**

*FROM Speciality s*

*JOIN Groupe g ON s.Name = g.spec*

*WHERE g.Id = :group\_id;*

**4.6 Управление уведомлениями**

Функция управления уведомлениями позволяет создавать и управлять уведомлениями, которые необходимы для информирования пользователей о различных событиях и задачах.

*// Определение переменных*

*TDateTime currentTime = Date(), InputTime, InputTimeS, SetTime = TDateTime(0, 0, 0.5, 0), SitTime;*

*String currentStr, InputStr, InputS, SitStr, hoop;*

*double you = i;*

*// Закрываем и очищаем запрос*

*ADOQuery1->Close();*

*ADOQuery1->SQL->Clear();*

*ADOQuery1->SQL->Add("USE Practic\_work");*

*ADOQuery1->SQL->Add("SELECT \* FROM Diary");*

*ADOQuery1->Open();*

*// Обработка записей в Diary*

*while (!ADOQuery1->Eof) {*

*InputTime = ADOQuery1->FieldByName("DataTime")->AsDateTime;*

*currentStr = DateTimeToStr(currentTime);*

*// Проверка и корректировка времени*

*if (RadioButton1->Checked && InputTime >= currentTime) if ((InputTime - jem) < currentTime) jem--;*

*InputTime -= jem;*

*InputStr = DateTimeToStr(InputTime);*

*if (InputStr.Length() > 10) InputStr.Delete(11, 19);*

*InputTime = StrToDateTime(InputStr);*

*// Сравнение текущего времени и времени события*

*if (currentTime == InputTime) {*

*InputTimeS = ADOQuery1->FieldByName("DataTime")->AsDateTime;*

*InputTimeS -= jem;*

*InputS = DateTimeToStr(InputTimeS);*

*if (InputS.Length() > 10) InputS.Delete(1, 11);*

*InputTimeS = StrToDateTime(InputS);*

*// Формирование строк уведомлений*

*if (piir == "" || mot >= 0) {*

*while (mot >= 0) {*

*if (RadioButton1->Checked) piir += DateTimeToStr(InputTimeS - (mot / 24.0)) + ' ';*

*else if (RadioButton2->Checked) piir += DateTimeToStr(InputTimeS + jem - ((mot / number) \* jem)) + ' ';*

*else if (RadioButton3->Checked) piir += DateTimeToStr(InputTimeS + jem - ((mot / number) \* jem)) + ' ';*

*mot--;*

*}*

*}*

*// Проверка и показ балуна*

*times = Time();*

*if (piir != "") {*

*hoop = piir.SubString(12, 8);*

*SitTime = StrToDateTime(hoop);*

*if (times > SitTime) {*

*piir.Delete(1, 20);*

*hoop = piir.SubString(12, 8);*

*if (piir != "") SitTime = StrToDateTime(hoop);}}*

*SitTime = SitTime - times;*

*SitStr = DateTimeToStr(SitTime);*

*if (SitStr.Length() > 10) SitStr.Delete(0, 11);*

*SitTime = StrToDateTime(SitStr);*

*if (SitTime <= SetTime) {*

*TrayIcon1->BalloonHint = ADOQuery1->FieldByName("Name")->AsString;*

*TrayIcon1->BalloonTimeout = 15000;*

*TrayIcon1->BalloonFlags = bfInfo;*

*TrayIcon1->ShowBalloonHint();*

*piir.Delete(1, 20);*

*i--;}}*

*ADOQuery1->Next();*

*if (i <= 0 || piir == "") {*

*Timer2->Enabled = false;*

*CheckBox1->Checked = false;}}*

**5. Заключение**

Результаты выполненной работы: В ходе выполнения работы была успешно разработана система управления студентами и преподавателями, включая функции добавления, редактирования и удаления записей. Реализованы запросы к базе данных для работы с группами и специальностями, а также разработаны механизмы уведомлений для напоминаний о событиях. Проведена интеграция всех компонентов системы и обеспечена ее стабильная работа с корректной обработкой данных.

Достижения целей и задач учебной практики: Цели и задачи учебной практики были достигнуты, включая:

* Разработка и оптимизация SQL-запросов для работы с таблицами студентов, групп и специальностей.
* Реализация пользовательского интерфейса для управления данными.
* Интеграция уведомлений о событиях с помощью таймеров и системного трея.
* Проведение тестирования и отладки системы для обеспечения ее функциональности и надежности.

Перспективы дальнейшего развития программного обеспечения: В перспективе планируется:

* Расширение функционала для более гибкого управления данными, включая возможность импорта и экспорта данных.
* Интеграция с другими системами и сервисами для расширения возможностей управления и анализа.
* Оптимизация производительности и улучшение пользовательского интерфейса на основе отзывов пользователей.
* Разработка мобильной версии приложения для обеспечения доступа к системе из любых устройств.

**Список использованных источников**

**6. Литература и материалы, использованные при разработке:**

**Книги и статьи:**

1. **"SQL для профессионалов"** – К. Грубер, [Издательство: Питер], 2018.
2. **"Разработка программного обеспечения: методы и практики"** – Д. Хоулс, [Издательство: БХВ-Петербург], 2017.
3. **"Основы работы с базами данных"** – М. Смирнов, [Издательство: ДМК Пресс], 2016.

**Документация и онлайн-ресурсы:**

1. **Официальная документация Microsoft SQL Server:** <https://docs.microsoft.com/en-us/sql/sql-server/>
2. **Официальная документация Delphi:** https://docwiki.embarcadero.com/RADStudio/Sydney/en/Main\_Page
3. **Ресурсы по программированию на C++:** https://cplusplus.com/doc/
4. **Форумы и сообщества разработчиков:**
   * **Stack Overflow:** <https://stackoverflow.com/>
   * **GitHub:** <https://github.com/>

**Учебные материалы:**

1. Лекции и методические материалы по базам данных и программированию, предоставленные в рамках учебной практики.
2. Примеры кода и учебные материалы из учебных пособий по Delphi и SQL, предоставленные преподавателями и учебными курсами.

Эти источники помогли в разработке системы, обеспечив глубокое понимание работы с базами данных, программирования на Delphi и созданию эффективных пользовательских интерфейсов.