# 



**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

ФГБОУ ВО «Брянский государственный технический университет» (БГТУ)

Политехнический колледж (ПК БГТУ)

09.02.03

**Курсовая работа**

**по ПМ 02 «Разработка и эксплуатация баз данных»**

**МДК 02.02 «Технология разработки и защиты баз данных»**

**Тема: «Контроль выполнения нагрузки преподавателей»**

**Пояснительная записка**

**ПКТУ. КР 00.000 ПЗ**

|  |  |
| --- | --- |
| Группа: | О-19-ПРО-4-СПО |
| Студент | Пайко С.А |
| Преподаватель |  |
| Дата |  |
| Оценка |  |

|  |
| --- |
| 2022 |

|  |
| --- |
| Шепотатьева С.С. |



**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

ФГБОУ ВО «Брянский государственный технический университет» (БГТУ)

Политехнический колледж (ПК БГТУ)

Утверждаю

Председатель цикловой комиссии

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«23» 12. 2021 г.

### Задание

**для курсовой работы**

**по модулю «Разработка и эксплуатация баз данных»**

**МДК 02.02 «Технология разработки и защиты баз данных»**

**Студент**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Тема работы:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Содержание и объем работы*

Курсовая работа состоит из пояснительной записки (30-40 листов) формата А4.

**Пояснительная записка**.

Титульный лист

Задание

Содержание

Введение

Реализация базы данных и серверной части информационной системы

Проектирование и разработка приложения

Заключение

Список используемых источников

Приложение

**ОГЛАВЛЕНИЕ**

[Задание 2](#_Toc105691542)

[**ВВЕДЕНИЕ** 4](#_Toc105691543)

[**1. ОСНОВНАЯ ЧАСТЬ** 5](#_Toc105691544)

[**1.1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ** 5](#_Toc105691545)

[**1.2. ВЫБОР СРЕДСТВ РАЗРАБОТКИ** 8](#_Toc105691546)

[**1.3. ВЫРАБОТКА ТРЕБОВАНИЙ И ОГРАНИЧЕНИЙ** 9](#_Toc105691547)

[**1.4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ** 10](#_Toc105691548)

[**1.4.1. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ** 11](#_Toc105691549)

[**1.4.2. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ** 12](#_Toc105691550)  
[**1.4.3. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ** 13](#_Toc105691551)

[**1.5. ПРОГРАММА РЕАЛИЗАЦИИ ПРОЕКТА БАЗЫ ДАННЫХ В MS SQL SERVER** 16](#_Toc105691552)

[**1.6. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР** 17](#_Toc105691553)

[**1.7. РАЗРАБОТКА ТРИГГЕРОВ ДЛЯ ПОДДЕРЖКИ СЛОЖНЫХ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ В БАЗЕ ДАННЫХ** 18](#_Toc105691554)

[**1.8. ЗАПРОСЫ** 20](#_Toc105691555)

[**1.9. ПРЕДСТАВЛЕНИЯ** 21](#_Toc105691556)

[**2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ** 22](#_Toc105691557)

[**2.1. РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ ПРИЛОЖЕНИЯ** 22](#_Toc105691558)

[**2.2. ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ** 23](#_Toc105691559)

[**2.3. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ** 26](#_Toc105691560)

[**ЗАКЛЮЧЕНИЕ** 27](#_Toc105691561)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 29](#_Toc105691562)

[**ПРИЛОЖЕНИЯ** 31](#_Toc105691563)

[**ПРИЛОЖЕНИЕ А (ЛИСТИНГ ПРОГРАММЫ)** 31](#_Toc105691564)

# **ВВЕДЕНИЕ**

В современном обществе огромное количество организаций используют базы данных для автоматизации процессов, быстрой и удобной обработки информации. Они помогают избавиться от ошибок в работе, повышают уровень надёжности хранения данных, повышают производительность, а также помогают предотвращать избыточность данных. Решение широкого круга задач во множестве сфер деятельности человека невозможно без управляемых баз данных.

Целью моей курсовой работы является разработка и проектирование базы данных «Контроль выполнения нагрузки преподавателей».

Для достижения поставленной цели необходимо поставить и решить следующие задачи:

* изучить предметную область;
* собрать необходимую информацию о ней;
* создать таблицы и запросы на основе данных;
* создать хранимые процедуры, триггеры и представления на основе данных;
* составить нормализацию отношений;
* автоматизировать базу данных проката самокатов;
* исследовать основные понятия.

Разрабатываемая база данных может быть использована для контроля выполнения нагрузки преподавателей в учебных заведениях. В ней можно будет отслеживать текущих преподавателей и проводимые ими занятия. В будущем созданную базу данных можно будет расширить для использования в других целях и задачах.

# **1. ОСНОВНАЯ ЧАСТЬ**

## **1.1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Качество подготовки преподавателей учебных заведений, а также эффективность использования их научно-педагогического потенциала в определенной степени зависят от уровня организации учебного процесса. Расписание занятий является одной из основных составляющих образовательной программы.

Чтобы упорядочить педагогическую работу и последовательность подачи знаний студентам, нужно учесть множество различных нюансов и тонкостей. Грамотно составленное расписание:

* регламентирует трудовой ритм;
* способствует равномерному распределению учебной нагрузки учащихся;
* обеспечивает работоспособность студентов и условия для их разносторонней деятельности;
* влияет на творческую отдачу преподавателей.

Подготовить идеальное расписание невозможно. Однако этот процесс можно, как минимум, значительно упростить и ускорить за счёт использования специализированного программного обеспечения.

В рамках данной курсовой работы рассмотрена предметная область «Контроль выполнения нагрузки преподавателей». Главная цель – создание такого программного обеспечения, которое бы позволило быстро составлять эффективный учебный план для преподавателей вузов.

Рассмотрим структурно – функциональную модель в нотации IDEF0 представленной на рисунке 1.1.

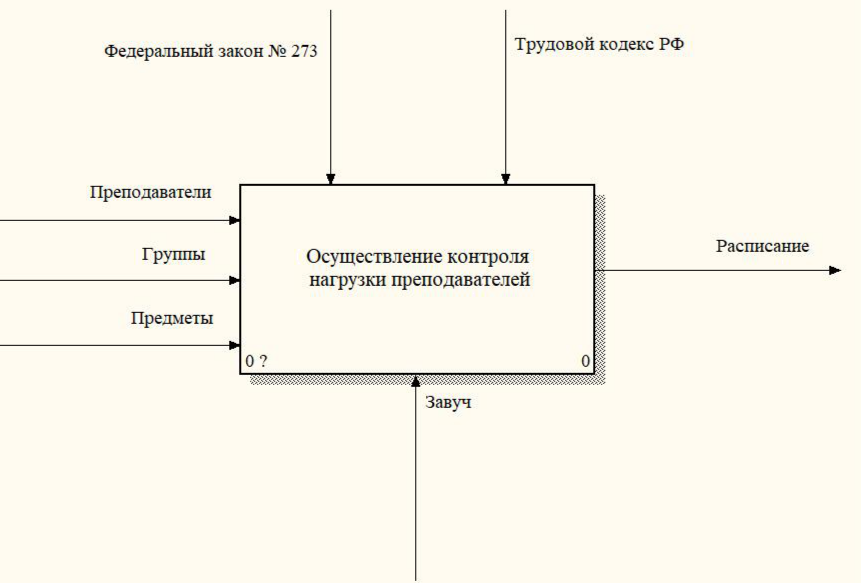


Рисунок 1.1 – Контекстная диаграмма.

Углубимся в процесс создания расписания для преподавателей.  
Созданием расписания для вуза обычно занимается заведующий учебной частью.

Для создания расписания нужно получить следующие данные:

* Данные о преподавателях
* Данные об обучаемых группах
* Данные о преподаваемых предметах

Получив нужные данные, можно преступить к созданию расписания. Декомпозиция контекстной диаграммы представлена на рисунке 1.2.

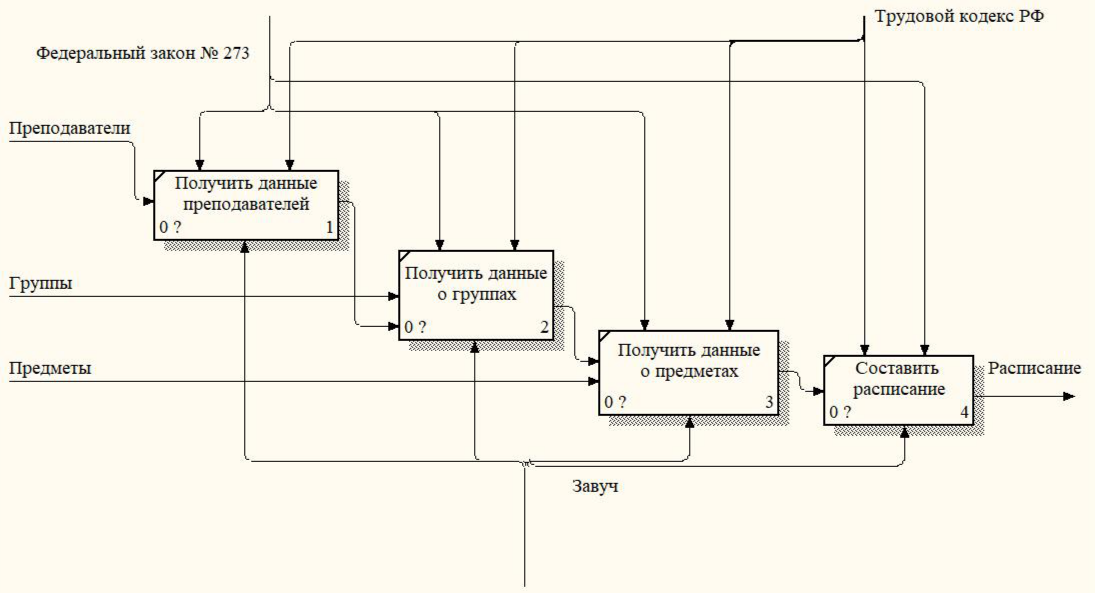


Рисунок 1.2 – Диаграмма декомпозиции

## **1.2. ВЫБОР СРЕДСТВ РАЗРАБОТКИ**

Разработка программного обеспечения выполняется на Visual Studio Community 2022.

Microsoft Visual Studio – это интегрированная среда разработки (IDE) от Microsoft. Используется для разработки компьютерных программ, а также веб-сайтов, веб-приложений, веб-сервисов и мобильных приложений. Visual Studio использует платформы разработки программного обеспечения Microsoft, также как Windows API, Windows Forms, Windows Presentation Foundation, Windows Store и Microsoft Silverlight. Может создавать как машинный код, так и управляемый код. В проекте используется Microsoft Visual Studio из-за расширенных библиотек, простоты и гибкости.

Разработка базы данных выполняется на SQL Server Management Studio 2019.

SQL Server – это система управления реализационными базами данных. Основной используемый язык запросов Transact-SQL. Программа используется в данном проекте из-за простоты использования, данная СУБД поддерживает работу с другими продуктами Microsoft.

## **1.3. ВЫРАБОТКА ТРЕБОВАНИЙ И ОГРАНИЧЕНИЙ**

Основные ограничения целостности:

1. Числовые атрибуты:

* Числовые атрибуты-идентификаторы(«ID») должны быть уникальными
* Числовые атрибуты должны быть больше или равны нулю
* Все числовые атрибуты должны быть заполнены
* «Рабочие часы» в таблице «Преподаватели» не должны заполняться вручную пользователем

1. Символьные атрибуты:

* Все символьные атрибуты должны быть заполнены
* Атрибут «Фамилия Имя Отчество»(«FullName») в таблице «Преподаватели» должен быть уникальным

## **1.4. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ**

База данных «Контроль выполнения нагрузки преподавателей» должна быть проведена таким образом, чтобы реализовать все требования к безопасности и целостности данных в проектируемой базе данных. На стадии проектирования и разработки базы данных осуществляется тщательный разбор процессов, происходящих с данными для дальнейшего грамотного построения концептуальной схемы и схемы данных.

### **1.4.1. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

После анализа предметной области можно выделить главные сущности: преподаватели, пары и расписание. «Преподаватели» ведут «Пары», из которых составляется «Расписание».

Это можно увидеть на ER-модели представленной на рисунке 1.3

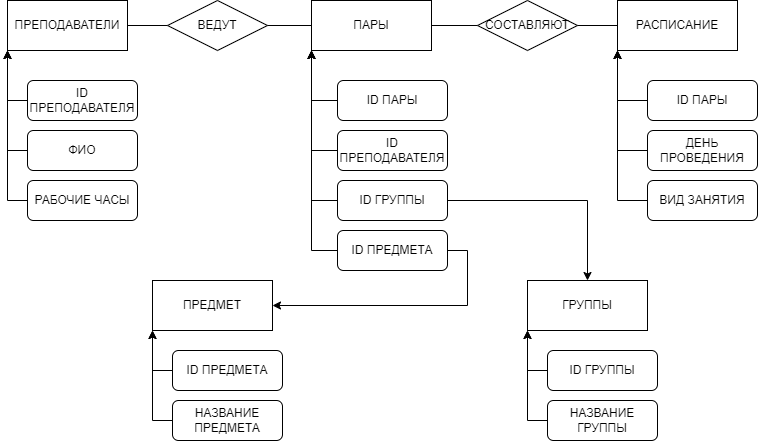
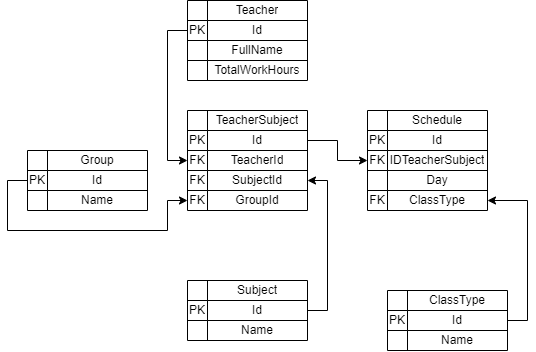


Рисунок 1.3 – ER модель

### **1.4.2. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

Логическое проектирование баз данных помогает определить и передать требования к информации предметной области. При создании логической схемы базы данных описывается каждый фрагмент информации, который необходимо отслеживать, и взаимосвязи между этими фрагментами информации, а также законы предметной области, которые управляют этими фрагментами информации.

Рассмотрим связи между выявленными сущностями:

  
Рисунок 1.4 – Логическая модель

## **1.4.3. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

На этапе физического проектирование базы данных происходит оптимизация производительности, при этом обеспечивается целостность данных за счет исключения ненужной избыточности данных. Во время физического проектирования преобразуются: объекты в таблицы, экземпляры в строки, а атрибуты в столбцы. После завершения логического проектирования нашей базы данных мы теперь перемещаемся к физическому дизайну

Таблица 1.1 – Атрибуты таблицы «Teacher»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Размер | Описание |
| Id | int | авто | Номер преподавателя |
| FullName | nchar | 60 | ФИО преподавателя |
| TotalWorkHours | int | авто | Количество запланированных рабочих часов |

Таблица 1.2 – Атрибуты таблицы «Subject»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Размер | Описание |
| Id | int | авто | Номер предмета |
| Name | nchar | 30 | Название предмета |

Таблица 1.3 – Атрибуты таблицы «Group»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Размер | Описание |
| Id | Int | Авто | Номер группы |
| Name | nchar | 30 | Название группы |

Таблица 1.4 – Атрибуты таблицы «TeacherSubject»

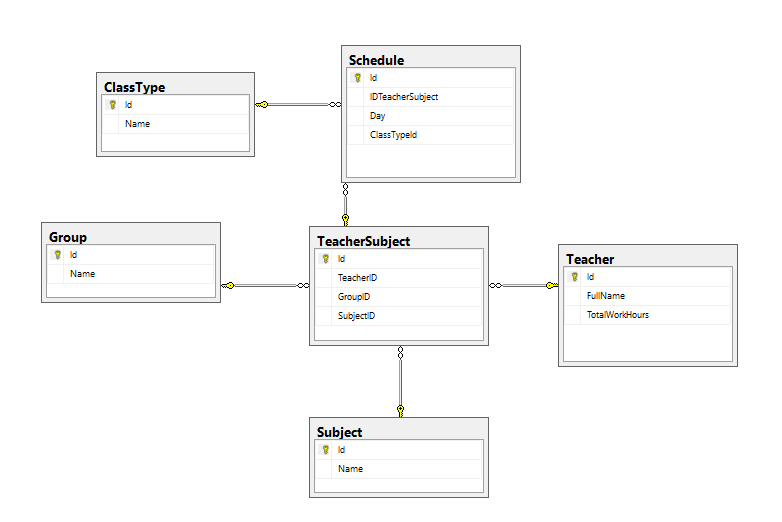
|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Размер | Описание |
| Id | int | авто | Номер пары |
| TeacherId | int | авто | Номер преподавателя |
| GroupId | int | авто | Номер группы, |
| SubjectId | int | авто | Номер предмета |

Таблица 1.5 – Атрибуты таблицы «Schedule»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Размер | Описание |
| Id | int | авто | Номер расписания |
| IDTeacherSubject | int | авто | Номер пары |
| Day | DateTime | авто | Дата и время проведения пары |
| ClassTypeId | int | авто | Вид пары: лекция или лабораторная работа |

Таблица 1.6 – Атрибуты таблицы «ClassType»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип | Размер | Описание |
| Id | Int | Авто | Номер вида пары |
| Name | nchar | 30 | Название |

  
Рисунок 1.5 – Физическая модель базы данных

## **1.5. ПРОГРАММА РЕАЛИЗАЦИИ ПРОЕКТА БАЗЫ ДАННЫХ В MS SQL SERVER**

Программа реализации проекта базы данных выполнена с помощью операторов языка SQL: USE, CREATE, DROP, UPDATE, ALTER, INSERT.

Текст программы создания базы данных приведён в приложении A. Для спроектированной базы данных средствами СУБД Microsoft SQL Server 2019 построена диаграмма, которая приведена в пункте 1.4.3.

Текст программы ввода текстовых данных приведен в приложении А.

## **1.6. РАЗРАБОТКА ХРАНИМЫХ ПРОЦЕДУР**

1. Процедура считает количество пар записанных в расписание

create procedure CountClasses

as

begin

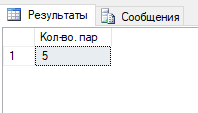
select count(Id) as [Кол-во. пар] from Schedule  
end  
  


Рисунок 1.6 – Результат выполнения CountClasses

1. Процедура считает количество пар у конкретного преподавателя

create procedure CountClassesForTeacher @id int

as

begin

select (TotalWorkHours / 2) as [Кол-во. пар у ], FullName as [Преподаватель] from Teacher where Id = @id

end

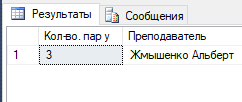
****

Рисунок 1.7 – Результат выполнения CountClassesForTeacher

1. Процедура считает количество пар в определенный день

create procedure CountClassesForDay @date DateTime

as

begin

select count(id) from Schedule where [Day] = @date

end

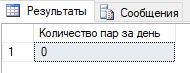


Рисунок 1.8 – Результат выполнения CountClassesForDay

## **1.7. РАЗРАБОТКА ТРИГГЕРОВ ДЛЯ ПОДДЕРЖКИ СЛОЖНЫХ ОГРАНИЧЕНИЙ ЦЕЛОСТНОСТИ В БАЗЕ ДАННЫХ**

1. Написать триггер на добавление для таблицы «Schedule». Триггер должен добавлять 2 рабочих часа для преподавателя в таблице «Teacher» за каждую пару, которую он будет должен провести по расписанию

create trigger u\_Schedule on Schedule

for insert

as

begin

update Teacher set TotalWorkHours += 2 where Id = (select TeacherId from TeacherSubject where Id = (select IDTeacherSubject from inserted))

end

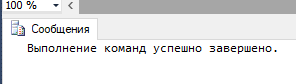


Рисунок 1.9 – Результат выполнения триггера u\_Schedule

1. Написать триггер на изменение таблицы «Teacher». Триггер должен отменять изменения, затрагивающие столбец «TotalWorkHours» и выводить сообщение: «Прямое изменение Рабочих часов запрещено»

create trigger u\_Teacher on Teacher

for update

as

begin

if update(TotalWorkHours)

begin

print ‘Прямое изменение Рабочих часов запрещено’

rollback transaction

end

end

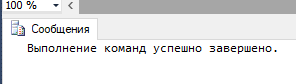


Рисунок 1.10 – Результат выполнения триггера u\_Teacher

1. Написать триггер на удаление для таблицы «Schedule». Триггер должен отнимать 2 рабочих часа для преподавателя в таблице «Teacher» за каждую пару, удаленную из расписания

create trigger d\_Schedule on Schedule

for delete

as

begin

update Teacher set TotalWorkHours -= 2 where Id = (select TeacherId from TeacherSubject where Id = (select IDTeacherSubject from deleted))

update Teacher set TotalWorkHours = 0 where TotalWorkHours < 0

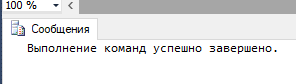
end  


Рисунок 1.11 – Результат работы триггера d\_Schedule

## **1.8. ЗАПРОСЫ**

1. Вывести все пары, проводимые Жмышенко Альбертом

select \* from TeacherSubject

where TeacherId = (select Id from Teacher where FullName = N'Жмышенко Альберт')

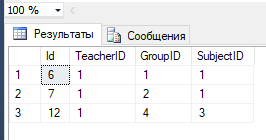


Рисунок 1.12 – Результат выполнения запроса

1. Вывести все дни, в которые планируется провести лекционные занятия

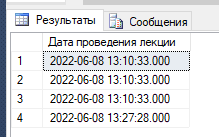
select [Day] as [Дата проведения лекции] from Schedule where ClassTypeId = (select Id from ClassType where Name = N'Лекция') 

Рисунок 1.13 – Результат выполнения запроса

## **1.9. ПРЕДСТАВЛЕНИЯ**

1. Вывести имена преподавателей, имеющих запланированные пары

create view bruv(FullName)

as

select FullName from Teacher where TotalWorkHours > 0

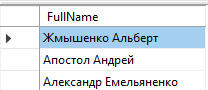


Рисунок 1.14 – Результат работы представления

1. Вывести пары и дни, внесённые в расписание

create view bruv2(IDTeacherSubject, [Day])

as

select IDTeacherSubject, [Day] from Schedule

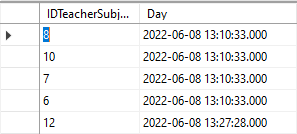


Рисунок 1.15 – Результат работы представления

# **2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРИЛОЖЕНИЯ**

## **2.1. РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ ПРИЛОЖЕНИЯ**

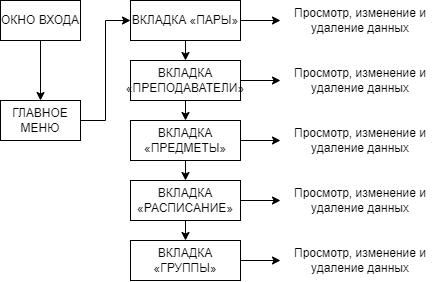


Рисунок 2.1 – Схема приложения

1. «Окно входа». На данном модуле происходит проверка регистрационных данных пользователя и последующий вход в приложение с переходом на модуль «Главное окно».
2. Окно «Главное меню». В этом модуле находится основная рабочая часть всего приложения, имеется возможность перехода с одной вкладки на другую, для выбора необходимых действий.

## **2.2. ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ**

На рисунке 2.2.1 представлена форма «Авторизация».

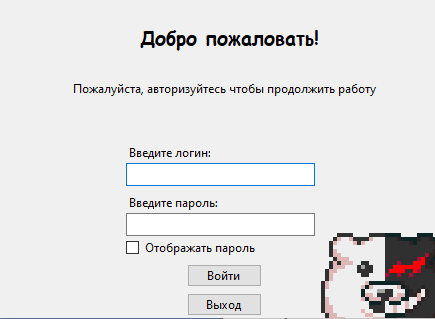


Рисунок 2.2.1 – Форма авторизации

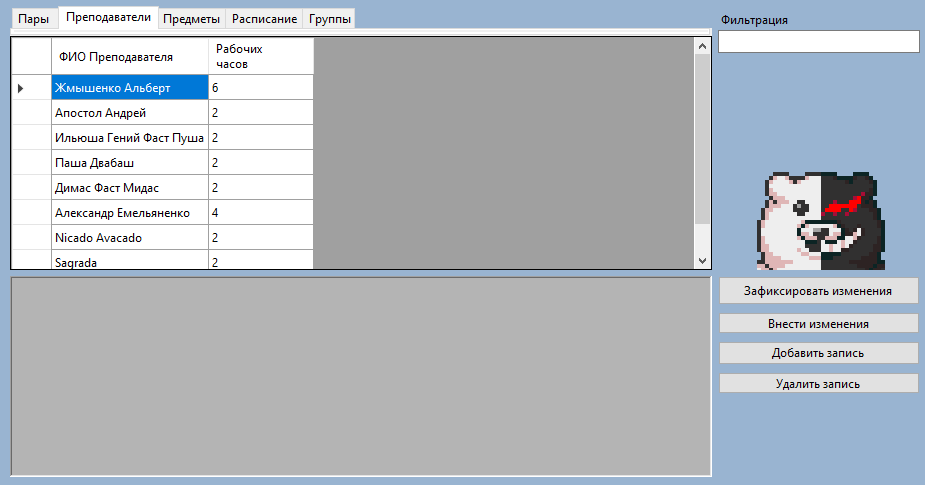
На рисунке 2.2.2 представлена форма «Главного окна» на вкладке «Преподаватели».

Рисунок 2.2.2 – Форма «Главного окна» на вкладке «Преподаватели»

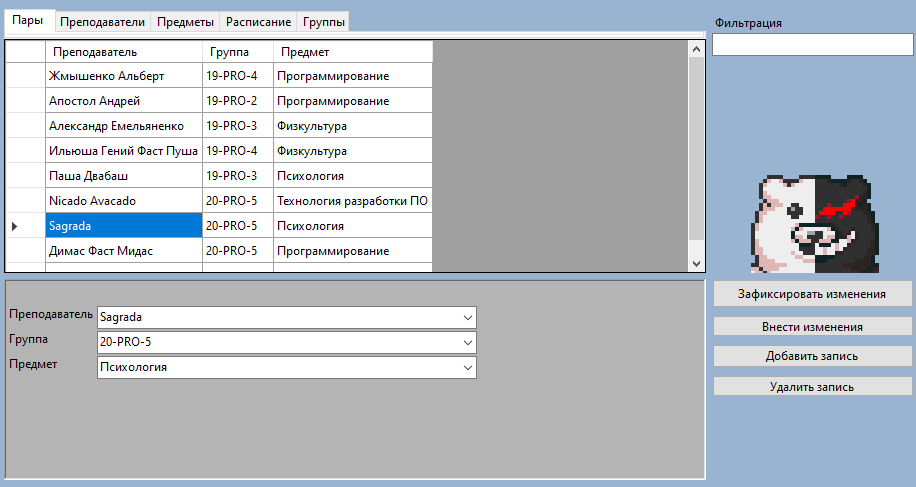
На рисунке 2.2.3 представлена форма «Главного окна» на вкладке «Пары». 

Рисунок 2.2.3 – Форма «Главного окна» на вкладке «Пары»

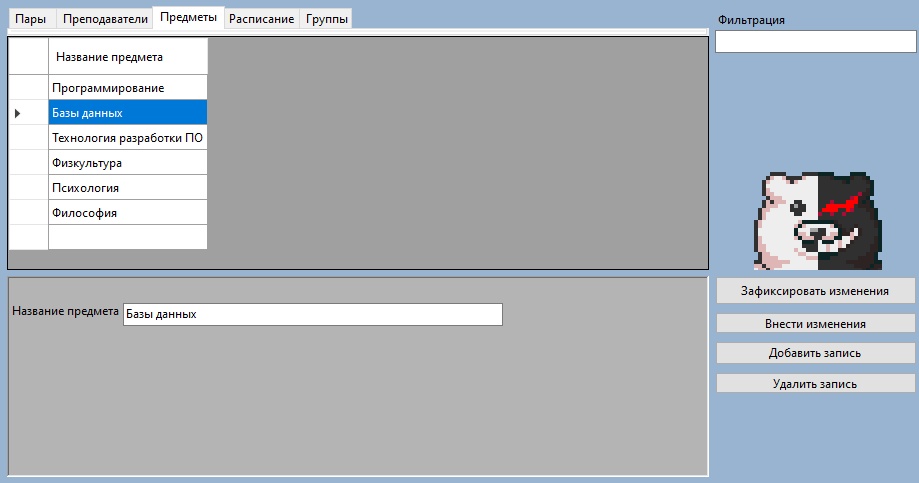
На рисунке 2.2.4 представлена форма «Главного окна» на вкладке «Предметы». 

Рисунок 2.2.4 – Форма «Главного окна» на вкладке «Предметы»

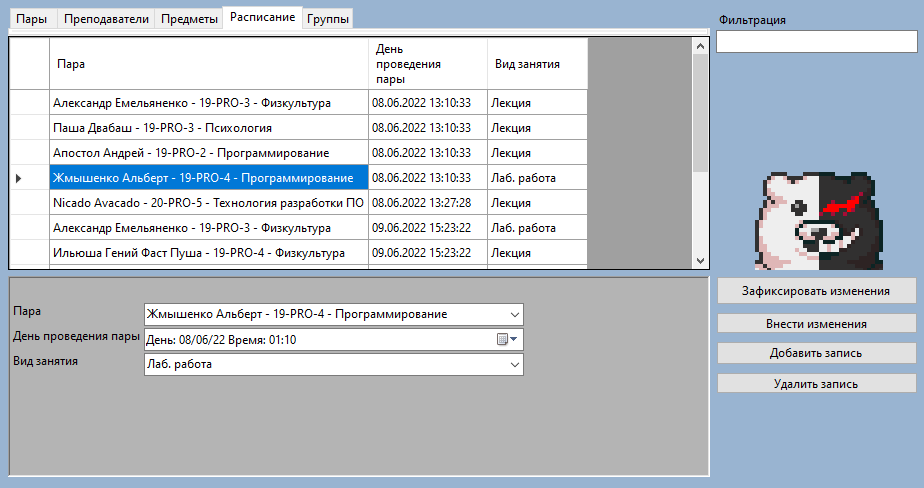
На рисунке 2.2.5 представлена форма «Главного окна» на вкладке «Расписание». 

Рисунок 2.2.5 – Форма «Главного окна» на вкладке «Расписание»

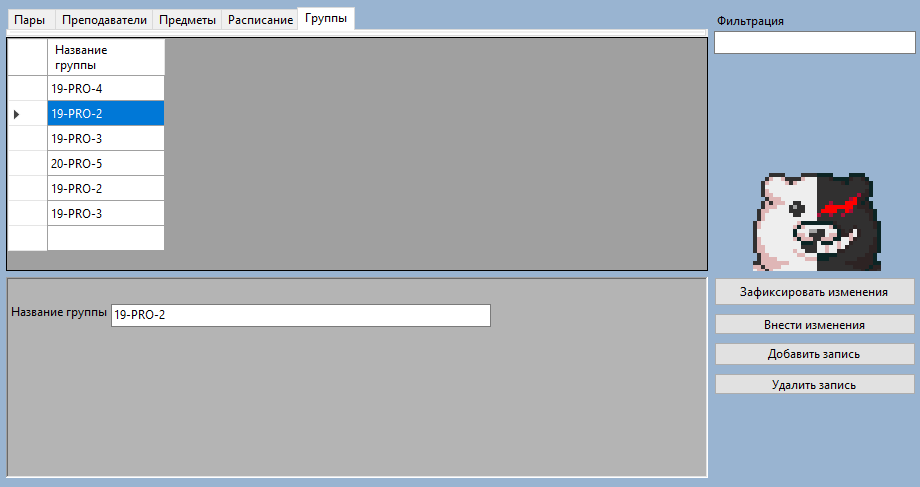
На рисунке 2.2.6 представлена форма «Главного окна» на вкладке «Группы». 

Рисунок 2.2.6 – Форма «Главного окна» на вкладке «Расписание»

## **2.3. ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ**

На рисунке 2.3.1 представлено сообщение о неверных входных данных.

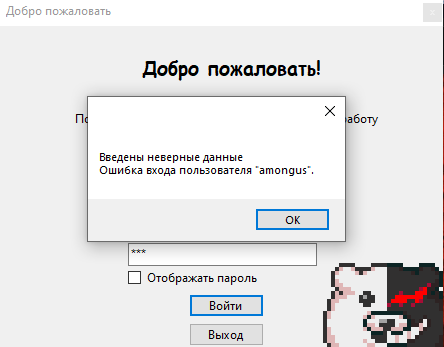
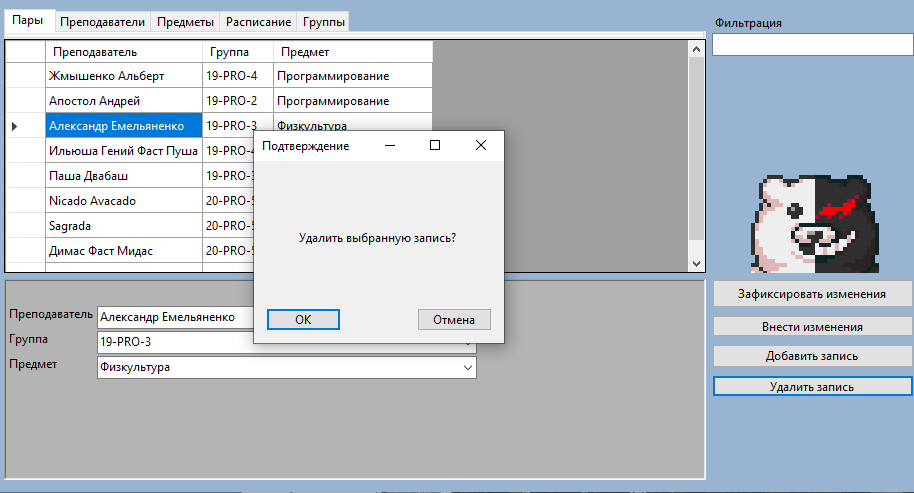


Рисунок 2.3.1 – Сообщение о неверных входных данных

На рисунке 2.3.2 представлено сообщение об удалении записи.

 Рисунок 2.3.3 – Сообщение об удалении записи

# **ЗАКЛЮЧЕНИЕ**

В процессе написания курсовой работы был проведен анализ предметной области контроля выполнения нагрузки преподавателей, после которого было принято решение о создании базы данных. Для достижения поставленной цели были изучены и использованы: система управления реляционными базами данных (СУБД) SQL Server Management Visual Studio.

В результате выполнение курсовой работы были решены следующие задачи: собран материал по предметной области и произведён его анализ, созданы таблицы и запросы на основе собранных данных, созданы хранимые процедуры, триггеры и представления, составлена нормализация отношений, автоматизирована база данных проката самокатов, спроектировано и разработано приложение, исследованы основные понятия.

В итоге разработанный программный продукт позволяет отслеживать информацию о прокате парах, преподавателях и их нагрузки.

Предполагается дальнейшее развитие и совершенствование разработанной базы данных, например, добавление функции создания отчётов.**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Joseph, J. Bambara SQL Server® Developer's Guide / Joseph J. Bambara, Paul R. Allen. - Москва: Мир, 2016. - 235 c.
2. MySQL руководство администратора. - М.: Вильямс, 2005. - 621 c.
3. Аткинсон MySQL. Библиотека профессионала / Аткинсон, Леон. - М.: Вильямс, 2014. - 624 c.
4. Биллиг В.A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В.A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c.
5. Гукин, Д. Для "чайников". Программирование на C / Д. Гукин. - М.: Вильямс, 2016. - 384 c.
6. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, 2017. - 480 c.
7. Кариев Ч.А. Разработка Windows-приложений на основе Visual C# [Электронный ресурс] : учебное пособие / Ч.А. Кариев. — Электрон. текстовые данные. — Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 768 c.
8. Конверс PHP 5 и MySQL. Библия пользователя / Конверс, др. Т. и. - М.: Вильямс, 2006. - 426 c.
9. Мартишин, С.А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и технолог / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, 2017. - 62 c
10. Мартишин, С.А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и техноло / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, 2018. - 61 c.
11. Оппель, Эндрю Дж. SQL. Полное руководство / Оппель Эндрю Дж.. - М.: Диалектика / Вильямс, 2016. - 902 c.
12. Павлова Е.А. Технологии разработки современных информационных систем на платформе Microsoft.NET [Электронный ресурс] / Е.А. Павлова. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 128 c.
13. Постолит Visual Studio .NET: разработка приложений баз данных / Постолит, Анатолий. - М.: СПб: БХВ, 2017. - 544 c.
14. Прайс, Джейсон Oracle Database 11g: SQL. Операторы SQL и программы PL/SQL / Джейсон Прайс. - М.: ЛОРИ, 2016. - 660 c.
15. Работа с MySQL, MS SQL Server и Oracle в примерах. В 2 ч.
16. Стружкин, Н.П. Базы данных: проектирование: Учебник для академического бакалавриата / Н.П. Стружкин, В.В. Годин. - Люберцы: Юрайт, 2016. - 477 c.
17. Троелсен, Эндрю Язык программирования C# 5.0 и платформа .NET 4.5 / Эндрю Троелсен. - М.: Вильямс, 2015. - 486 c.

учеб.-метод. пособие / С. С. Куликов, Е. Е. Фадеева. – Минск : БГУИР,

1. Яргер, Р.Дж. MySQL и mSQL: Базы данных для небольших предприятий и Интернета / Р.Дж. Яргер, Дж. Риз, Т. Кинг. - М.: СПб: Символ-Плюс, 2016. - 560 c.
2. Яргер, Р.Дж. MySQL и mSQL: Базы данных для небольших предприятий и Интернета / Р.Дж. Яргер, Дж. Риз, Т. Кинг. - М.: СПб: Символ-Плюс, 2015. - 560 c

# **ПРИЛОЖЕНИЯ**

## **ПРИЛОЖЕНИЕ А (ЛИСТИНГ ПРОГРАММЫ)**

create database TeacherWorkloadControl

use TeacherWorkloadControl;

go

create table [Subject]

(

Id int identity primary key,

Name nvarchar(30) not null,

);

create table ClassType

(

Id int identity primary key,

Name nvarchar(30) not null

);

create table [Group]

(

Id int identity primary key,

Name nvarchar(10) not null

);

create table Teacher

(

Id int identity primary key,

FullName nvarchar(60) not null,

TotalWorkHours int

);

create table TeacherSubject

(

Id int identity primary key,

TeacherID int foreign key references [Teacher](Id),

GroupID int foreign key references [Group](Id),

SubjectID int foreign key references [Subject](Id)

);

create table Schedule

(

Id int identity primary key,

ID\_Teacher\_Subject int foreign key references TeacherSubject(Id) not null,

[Day] DateTime not null,

ClassTypeId int foreign key references ClassType(Id) not null

)

## **ПРИЛОЖЕНИЕ Б (РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ)**

using CourseProject.Content.Models;

using CourseProject.Core;

using CourseProject.Core.Attributes;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Reflection;

using System.Threading;

using System.Windows.Forms;

namespace CourseProject

{

public partial class MainForm : Form

{

public const string ConnectionString = "Data Source=.\\SQLEXPRESS;Initial Catalog=TeacherWorkloadControl;";

private string \_userInfo = "Integrated Security = true;";

public string UserInfo

{

get => \_userInfo;

set

{

if (value != \_userInfo)

{

\_userInfo = value;

Connection = new SqlConnection(ConnectionString + \_userInfo);

}

}

}

public SqlConnection Connection { get; private set; }

public static MainForm instance { get; private set; }

public DataStorage<Model> TeacherSubject;

public DataStorage<Model> Teacher;

public DataStorage<Model> Subject;

public DataStorage<Model> Group;

public DataStorage<Model> Schedule;

public DataStorage<Model> ClassType;

public Type CurrentStorage;

public Dictionary<Type, DataStorage<Model>> Storages;

public Model CurrentModel { get; set; }

public Type lastKnownType;

public int currentId;

public Dictionary<Type, PropertyInfo[]> Infos = new Dictionary<Type, PropertyInfo[]>();

public MainForm()

{

instance = this;

InitializeComponent();

MainGrid.Columns.Clear();

Storages = new Dictionary<Type, DataStorage<Model>>();

Connection = new SqlConnection(ConnectionString + UserInfo);

TeacherSubject = new DataStorage<Model>(Connection, typeof(TeacherSubject));

Teacher = new DataStorage<Model>(Connection, typeof(Teacher));

Subject = new DataStorage<Model>(Connection, typeof(Subject));

Schedule = new DataStorage<Model>(Connection, typeof(Schedule));

Group = new DataStorage<Model>(Connection, typeof(Group));

ClassType = new DataStorage<Model>(Connection, typeof(ClassType));

Storages.Add(typeof(TeacherSubject), TeacherSubject);

Storages.Add(typeof(Teacher), Teacher);

Storages.Add(typeof(Subject), Subject);

Storages.Add(typeof(Schedule), Schedule);

Storages.Add(typeof(Group), Group);

Storages.Add(typeof(ClassType), ClassType);

foreach (var t in Storages.Keys)

{

Infos.Add(t, t.GetProperties());

}

ScrewTabControlMyHomiesHateTabControl();

PullDatas();

ShowDataFor(Teacher);

lastKnownType = typeof(Teacher);

}

private void PullDatas()

{

foreach(var storage in Storages.Values)

{

PullDataFor(storage);

}

}

private void ClearDatas()

{

foreach (var storage in Storages.Values)

storage.Clear();

}

private void ScrewTabControlMyHomiesHateTabControl()

{

tableControl.TabPages.Clear();

foreach (var kvp in Storages)

{

NameAttribute attr = kvp.Value.expectedType.GetCustomAttribute<NameAttribute>();

if (attr != null)

{

TabPage pg = new TabPage(attr.Name)

{

Name = kvp.Key.Name

};

tableControl.TabPages.Add(pg);

}

}

tableControl.Selected += TableControl\_Selected;

tableControl.SelectedIndex = 1;

}

private void TableControl\_Selected(object sender, TabControlEventArgs e)

{

TabControl control = (TabControl)sender;

int index = control.SelectedIndex;

string name = control.TabPages[index].Name;

Type t = Assembly.GetAssembly(typeof(Model)).GetTypes().Where(x => x.Name == name).First();

ShowDataFor(Storages[t]);

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void PullDataFor<T>(DataStorage<T> storage)

where T : Model

{

Connection.Open();

bool needsBullshitWorkaround = storage.expectedType.GetCustomAttribute(typeof(NotKeywordAttribute)) != null;

string typeName = storage.expectedType.Name;

string tableName = needsBullshitWorkaround ? "[" + typeName + "]" : typeName;

string query = @"select \* from " + tableName;

SqlCommand c = new SqlCommand(query, Connection);

SqlDataReader reader = c.ExecuteReader();

while (reader.Read())

{

var data = (T)Activator.CreateInstance(storage.expectedType);

PropertyInfo[] infos = Infos[storage.expectedType];

int num = 0;

foreach (PropertyInfo f in infos.Where(x => x.GetCustomAttribute(typeof(NameAttribute)) != null))

{

var d = reader[num];

ref var obj = ref data;

f.SetValue(obj, d);

num++;

}

data.id = (int)infos[0].GetValue(data);

storage.Add(data);

}

Connection.Close();

reader.Close();

}

private void button1\_Click(object sender, System.EventArgs e)

{

ShowDataFor(Teacher);

}

private void showTeacherSubject\_Click(object sender, System.EventArgs e)

{

ShowDataFor(TeacherSubject);

}

private void commitChanges\_Click(object sender, System.EventArgs e)

{

foreach (var kvp in Storages)

{

kvp.Value.Fixate();

}

ClearDatas();

PullDatas();

ShowDataFor(Storages[CurrentStorage]);

}

private void MainGrid\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

}

private void RefreshDisplay()

{

MainGrid.Columns.Clear();

ShowDataFor(Storages[CurrentStorage]);

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void deleteSelected\_Click(object sender, EventArgs e)

{

ConfirmDeletionForm f = new ConfirmDeletionForm();

f.ShowDialog(this);

if (f.DialogResult == DialogResult.OK)

{

for(int i = MainGrid.SelectedCells.Count-1; i >= 0; i--)

{

int rowIndex = MainGrid.SelectedCells[i].RowIndex;

int id = (int)MainGrid[0, rowIndex].Value;

PropertyInfo[] infos = Infos[Storages[CurrentStorage].expectedType];

var get = Storages[CurrentStorage].Find(x => (int)infos[0].GetValue(x) == id);

get.ToBeRemoved = true;

MainGrid.Rows.RemoveAt(MainGrid.SelectedCells[i].RowIndex);

MainGrid.Update();

KillContextPanel();

}

foreach (var kvp in Storages)

{

kvp.Value.Fixate();

}

ClearDatas();

PullDatas();

}

}

private void MainForm\_Deactivate(object sender, EventArgs e)

{

}

private void MainForm\_FormClosed\_1(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void MainGrid\_CellClick(object sender, DataGridViewCellEventArgs e)

{

int id = -999;

if (MainGrid.RowCount > 0)

{

object val = MainGrid[0, MainGrid.SelectedCells[0].RowIndex].Value;

currentId = MainGrid.SelectedCells[0].RowIndex;

id = val != null ? (int)val : -999;

}

CurrentModel = Storages[CurrentStorage].Find(x => x.id == id);

MakeContextPanel();

}

private void addEntry\_Click(object sender, EventArgs e)

{

if (contextPanel.Controls.Count <= 0)

{

MakeContextPanel();

return;

}

if(CurrentModel != null && Storages[CurrentModel.GetType()].Find(x => x == CurrentModel) != null)

{

MessageBox.Show("Данная запись уже существует");

return;

}

else

{

Storages[lastKnownType].Add(GetModel(ModelStyle.New));

RefreshDisplay();

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

foreach(DataGridViewRow row in MainGrid.Rows)

{

row.Visible = true;

if (row.Cells[0].Value == null)

continue;

var model = Storages[CurrentStorage].GetAll().FirstOrDefault(x => x.id == (int)row.Cells[0].Value);

if (model == null)

continue;

string text = model.ToString().ToLower();

if (!text.Contains(filterBox.Text.ToLower()))

row.Visible = false;

}

}

}

}