МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Московский государственный университет технологий и управления имени К.Г. Разумовского

(Первый казачий университет)

Университетский колледж информационных технологий

Специальность: 09.02.03 «Программирование в компьютерных системах»

КУРСОВОЙ ПРОЕКТ

Модуль ПМ.02. «Разработка и администрирование баз данных»

МДК.02.02 «Технология разработки и защиты баз данных»

|  |  |  |
| --- | --- | --- |
| на тему: « Программа учёта сдачи работ студентами колледжа» |  |  |
| **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**  **УКИТ.**09.02.03***.*2018.304.049ПЗ** |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  | |  |
|  | |  | |  |
| Группа | | П-304к | |  |
| Руководитель проекта | **Личная подпись**  (личная подпись) | | Ларионова Е.А. | | |
| Нормативный контроль | **Личная подпись**  (личная подпись) | | Гусева Е.Л. | | |
| Студент | **Личная подпись**  (личная подпись) | | Максимов И.А. | | |

Москва 2018

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc517908454)

[1 ОБЩАЯ ЧАСТЬ 4](#_Toc517908455)

[1.1 Назначение и цели создания системы 4](#_Toc517908456)

[1.2 Обзор и анализ предметной области 4](#_Toc517908457)

[1.3 Жизненный цикл базы данных 5](#_Toc517908458)

[1.4 Выбор и характеристика СУБД 6](#_Toc517908459)

[1.5 Выбор и характеристика среды разработки приложения 6](#_Toc517908460)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 8](#_Toc517908461)

[2.1 Постановка задачи 8](#_Toc517908462)

[2.2 Архитектура информационной системы 8](#_Toc517908463)

[2.3 Логическая модель базы данных 9](#_Toc517908464)

[2.4 Нормализация таблиц 9](#_Toc517908465)

[2.5 Описание таблиц 13](#_Toc517908466)

[2.6 Разработка приложения 16](#_Toc517908474)

[2.7 Инструкция пользователю 19](#_Toc517908475)

[ЗАКЛЮЧЕНИЕ 23](#_Toc517908476)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc517908477)

[ПРИЛОЖЕНИЕ А SQL скрипты на создание и заполнение базы данных, на создание учетных записей пользователей, на назначение привилегий 25](#_Toc517908478)

[ПРИЛОЖЕНИЕ Б Код программы 34](#_Toc517908479)

# **ВВЕДЕНИЕ**

Курсовой проект посвящен разработке информационной системы, которая бы обеспечивала учет сдачи работ студентами колледжа.

Основной целью являлось выполнение учебного плана и получение опыта и ряда профессиональных навыков и компетенций.

Каждый предмет имеет свой набор тем, а каждая тема закрепляется за студентом группы.

Аналогов программы учёта сдачи работ студентами колледжа нет.

Анализ показал, что аналоги не имеют хорошей графической составляющей и сильно нагружены посторонней информацией.

Задачи проекта:

1. ) анализ предметной области,
2. ) проектирование базы данных и удобное администрирование базы данных осуществляется посредством инструмента MySql Workbench,
3. ) разработка приложения средствами IDE Visual Studio на языке C#,
4. ) защита базы данных посредством использования системы ролей в СУБД MySql.

Система должна позволять пользователям удобно занять тему для студента из группы, или распределить темы в группе автоматически.

**1 ОБЩАЯ ЧАСТЬ**

**1.1 Назначение и цели создания системы**

Система разрабатывается с целью реализации учебного плана и получения разработчиком профессиональных навыков и компетенций.

Назначением системы является автоматизация процесса распределения тем между студентами.

**1.2 Обзор и анализ предметной области**

В соответствии с предметной областью система строится с учётом следующих особенностей:

* каждая тема в группе не должна дублироваться с другим студентом, если темы были распределены в автоматическом порядке и без вмешательства пользователя,
* в каждом предмете свои темы,
* один студент может занять одну или несколько тем по определенному предмету.

Выделим базовые сущности этой предметной области:

Темы. Атрибуты номеров – id, имя, id предмета. Темы принадлежат предметам.

Студенты. Атрибуты клиентов – id, фамилия, id группы. Студенты находятся в группах.

Занятые темы будем рассматривать как связь между студентом и темой.

Атрибуты занятой темы – id студента, id группы студента, id темы, id предмета темы, текущая оценка.

Система создаётся для обслуживания следующих групп пользователей:

* студент,
* преподаватель,
* работник учебной части,
* администратор.

Границы информационной поддержки пользователей:

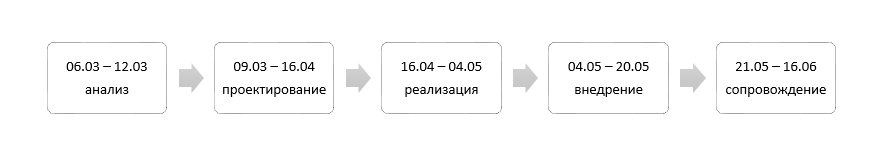
Функциональные возможности:

* ведение БД (запись, чтение, модификация),
* обеспечение логической непротиворечивости БД,
* обеспечение защиты данных от несанкционированного или случайного доступа,
* реализация наиболее часто встречающихся запросов в готовом виде,
* предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

Готовые запросы:

* получение списка, удаление, добавление занятых тем,
* получение списка, удаление, добавление студентов в группах,
* получение списка, удаление, добавление тем в предметах,
* получение списка, удаление, добавление учебного плана,
* получение списка, удаление, добавление предметов,
* получение списка, удаление, добавление групп.

**1.3 Жизненный цикл базы данных**

Рисунок 1 – Схема жизненного цикла базы данных

На этапе анализа (06.03 - 12.03) происходит анализ предметной области, выявление главных сущностей и пользователей, формулирование задачи, определение требований.

На этапе проектирования (09.03 - 16.04) происходит разработка концептуальной, логической модели базы данных, приведение ее к нормальным формам.

На этапе реализации (16.04 - 04.05) происходит реализация физической модели базы данных на основе логической модели, разрабатываются хранимые процедуры и отлаживаются основные и критические действия в базе данных.

На этапе внедрения (04.05 - 20.05) происходит опытная эксплуатационное базы данных, приобретение навыков работы персоналом, выявление и устранение выделенных недочетов системы, сдача в промышленную эксплуатацию.

Этап сопровождения (21.05 - 16.06) происходит во время эксплуатации и в ее ходе осуществляется анализ работы системы, исправление ошибок и недоработок, оформление требований и разработка планов по модернизации и расширению системы.

## 1.4 Выбор и характеристика СУБД

Для данной БД была выбрана СУБД MySql, поскольку она является бесплатно-распространяемой, Open Source, реляционно-объектной СУБД, то есть поддерживает пользовательские объекты и их поведение, включая типы данных, функции, операции, домены и индексы.

Сильными сторонами MySql считаются:

* высокопроизводительные и надежные механизмы транзакций и репликации,
* использование объектно-реляционного подхода,
* возможность работы с BigData (максимальный размер таблицы 32 ТБ),
* расширяемая система встроенных языков,
* поддержка целостности данных в соответствии со стандартом ANSI-SQL:2008 и отвечает требованиям ACID,
* открытый исходный код.

**1.5 Выбор и характеристика среды разработки приложения**

Для разработки приложения была выбрана среда C# и фреймворк Visual Studio, поскольку она позволяет писать высокопроизводительные приложения, а Visual Studio обеспечивает наилучшее графическое API и возможность взаимодействия приложения с базами данных. Также C# позволяет программисту работать на довольно низком уровне, это помогает эффективно контролировать приложение в критических моментах. Данная среда позволяет программисту выбрать уровень абстракции на котором ему будет писать быстрее и удобнее в зависимости от предметной области и поставленных задач.

Достоинства среды:

* высокая производительность,
* поддержка различных стилей программирования,
* перегрузка операторов позволяет кратко записывать выражения над пользовательскими типами в естественной алгебраической форме,
* широкий выбор плагинов позволяет быстро внедрить необходимые технологии в свой проект,
* возможность встраивания предметно-ориентированных языков программирования в основной код.

**2 СПЕЦИАЛЬНАЯ ЧАСТЬ**

**2.1 Постановка задачи**

Спроектировать средствами MySql базу данных и создать средствами C# приложение, позволяющее распределить студентам темы курсовых проектов в выбранной группе по заданному предмету таким образом, чтобы ни одна тема в пределах группы не повторилась. Распределение может производиться в 2х режимах: ручном и автоматическом. Автоматический путь предполагает распределение без участия пользователя, ручной – пользователем.

Тема предусматривает расширение по желанию автора. Обеспечить защиту базы данных и приложения аутентификацией.

**2.2 Архитектура информационной системы**

В данном проекте используется архитектура клиент-сервер с активным сервером, что означает, что часть бизнес логики будет выполняться на сервере посредством хранимых процедур. Схема данной архитектуры изображена на рисунке (Рисунок 2).

Достоинства:

* отсутствие дублирования кода программы-сервера программами-клиентами,
* так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются,
* все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только пользователям с соответствующими правами доступа.

Недостатки:

* неработоспособность сервера может сделать неработоспособной всю вычислительную сеть. Неработоспособным сервером следует считать сервер, производительности которого не хватает на обслуживание всех пользователей, а также сервер, находящийся на ремонте, профилактике и т. п.,
* поддержка работы данной системы требует отдельного специалиста – системного администратора,
* высокая стоимость оборудования.

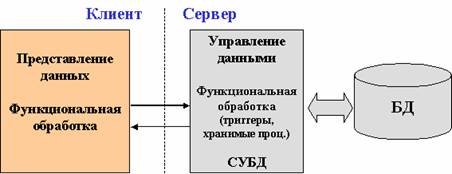


Рисунок 2 – Схема архитектуры клиент-сервер с активным сервером

**2.3 Логическая модель базы данных**

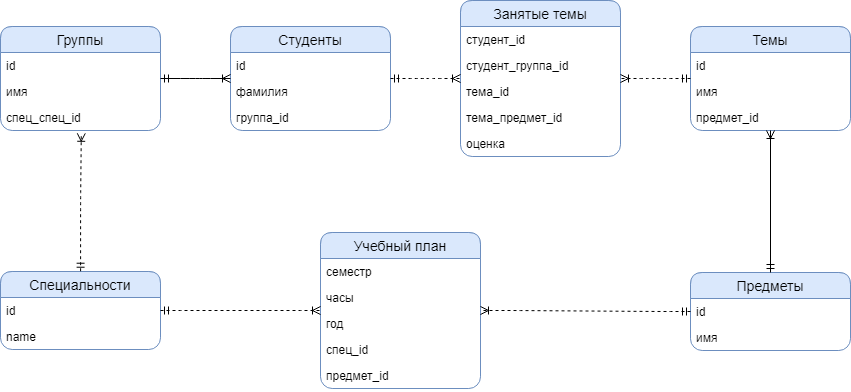


Рисунок 3 – Диаграмма базы данных резервирования номеров в гостиницах

**2.4 Нормализация таблиц**

1НФ. Отношения данной схемы не нарушают 1НФ, т.к. все атрибуты далее неделимы.

2НФ. В нашем случае составные первичные ключи имеют отношения:

* занятые темы – студент\_id, студент\_группа\_id, тема\_id, тема\_предмет\_id,
* учебный план – семестр, год, спец\_id, предмет\_id,
* группы – id, спец\_спец\_id.

Не ключевые атрибуты этих отношений функционально полно зависят от первичных ключей.

3НФ. Отношения данной схемы не нарушают 3НФ, т.к. не существуют транзитивных зависимостей между ключом и не ключевыми атрибутами. Отношения данной схемы не нарушают 3НФ, т.к. атрибуты не имеют функциональных зависимостей.

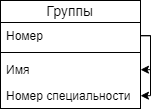


Рисунок 4 – Функциональные зависимости таблицы ГРУППЫ

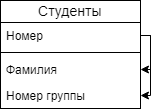


Рисунок 5 – Функциональные зависимости таблицы СТУДЕНТЫ



Рисунок 6 – Функциональные зависимости таблицы ЗАНЯТЫЕ ТЕМЫ

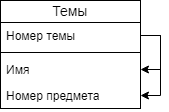


Рисунок 7 – Функциональные зависимости таблицы ТЕМЫ

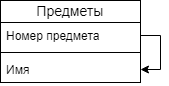


Рисунок 8 – Функциональные зависимости таблицы ПРЕДМЕТЫ

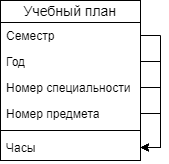


Рисунок 9 – Функциональные зависимости таблицы УЧЕБНЫЙ ПЛАН

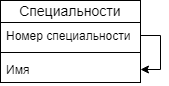


Рисунок 10 – Функциональные зависимости таблицы СПЕЦИАЛЬНОСТИ

**Определение дополнительных ограничений целостности**

Значения всех числовых атрибутов – больше 0 (или null, если атрибут

необязателен).

**Описание групп пользователей и прав доступа**

Незарегистрированные пользователи не имеют доступа к базе данных.

Студенты, просматривающие базу данных: имеют доступ по чтению к отношениям:

* группы,
* студенты,
* занятые темы,
* темы,
* предметы,
* учебный план,
* специальности.

Пользователи, вошедшие под администратором, преподавателем или работником учебной части имеют доступ по записи и чтению к отношениям:

* группы,
* студенты,
* занятые темы,
* темы,
* предметы,
* учебный план,
* специальности.

Пользователи, вошедшие под администратором, имеют доступ по записи и чтению к отношениям:

* группы,
* студенты,
* занятые темы,
* темы,
* предметы,
* учебный план,
* специальности,
* пользователи.

Доступ по записи предоставляется отношениям к Брони и Брони – Услуги.

**2.5 Описание таблиц**

Таблица 1 – Схема отношения ГРУППЫ (Groups)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер группы | Id | int(11) | Первичный ключ, Auto increment, обязательное поле |
| Имя группы | Name | varchar(45) | Обязательное поле |
| Номер специальности | Spec\_spec\_id | int(11) | Первичный ключ, обязательное поле |

Таблица 2 – Схема отношения СТУДЕНТЫ (Students)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер студента | id | int(11) | первичный ключ, обязательное поле |
| Фамилия | Surname | Varchar(45) | обязательное поле |
| Номер группы | Group\_id | Int(11) | Первичный ключ, обязательное поле |

Таблица 3 – Схема отношения ТЕМЫ (Themes)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер темы | Id | int(11) | первичный ключ |
| Наименование | Name | varchar(45) | обязательное поле |
| Продолжение таблицы 3 | | | |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер предмета | Subject\_id | Int(11) | Обязательное поле |

Таблица 4 – Схема отношения ЗАНЯТЫЕ ТЕМЫ (Engaged\_hemes)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер студента | Students\_id | Int(11) | Первичный ключ |
| Номер группы | Students\_Groups\_id | Int(11) | Первичный ключ |
| Номер темы | Themes\_id | Int(11) | Первичный ключ |
| Номер предмета | Themes\_Subjects\_id | Int(11) | Первичный ключ |
| Оценка | Mark | Int(11) | Обязательное поле |

Таблица 5 – Схема отношения ПРЕДМЕТЫ (Subjects)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер | Id | Int(11) | Первичный ключ |
| Наименование | name | Varchar(45) | Обязательное поле |

Таблица 6 – Схема отношения СПЕЦИАЛЬНОСТИ (Specialties)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер | id | Int(11) | Первичный ключ |
| Наименование | name | Varchar(45) | Обязательное поле |
| Код | code | Varchar(45) | Внешний ключ |

Таблица 7 – Схема отношения УЧЕБНЫЙ ПЛАН (Edu\_plan)

|  |  |  |  |
| --- | --- | --- | --- |
| Содержание поля | Имя поля | Тип, длина | Примечания |
| Номер семестра | Semester\_number | Int(11) | Первичный ключ |
| Часы | Hours | Int(11) | Обязательное поле |
| Годы | Year | Int(11) | Первичный ключ |
| Номер специальности | Specs\_id | Int(11) | Первичный ключ |
| Номер предмета | Subjects\_id | Int(11) | Первичный ключ |

2. 6. **Разработка приложения**
      1. **Структурная схема приложения**

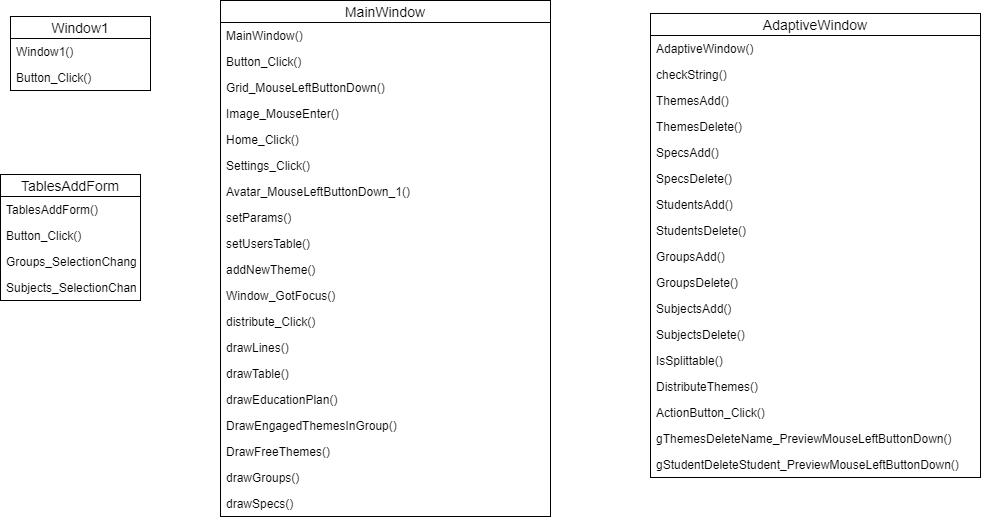
****

Рисунок 11 – UML диаграмма классов.

* + 1. **Функции приложения**

Резервирование выбранной свободной темы для определённого студента – распределение может производиться в 2х режимах: ручном и автоматическом. Автоматический путь предполагает распределение без участия пользователя, ручной – пользователем;

Формирование отчёта в PDF формате – формирование списка свободных тем по заданному предмету в заданной группе, а также формирование списка занятых тем по заданному предмету в заданной группе.

Поддержка БД в актуальном состоянии;

Отображение студентов, групп, предметов, занятых тем, учебного плана

и специальностей из-под программы;

Обеспечение защиты.

* + 1. **Решение главной задачи проекта**
       1. **Отображение всех занятых тем у студентов во всех группах:**

Show\_engaged\_themes();

Из соединения таблиц students, themes, groups, subjects получаются текстовые ответы с информацией о занятых темах в предметах у студентах в группах.

* + - 1. **Выбор свободной темы производится вызовом следующих функций:**

void StudentsAdd();

Обновляет результирующую таблицу с учетом последних изменений всех фильров.

**Резервирование свободного номера производится вызовом функции:**

void distribute\_Click(object sender, RoutedEventArgs e);

Выполняется добавление брони через вызов хранимой процедуры

public AdaptiveWindow(string connStr, string type, string action)

**Фрагмент приложения, где показана область программы с отображенными занятыми темами:**

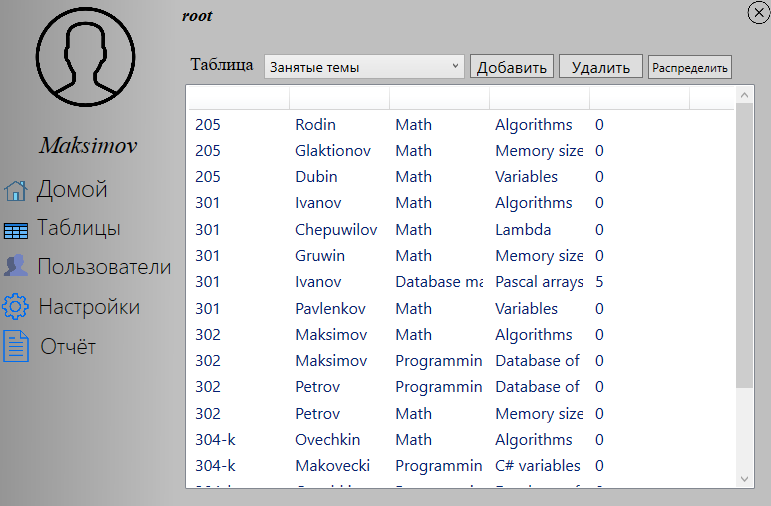


Рисунок 12 – Форма «Занятые темы»

* + - 1. **2 Фрагмент приложения, где отображено окно добавления темы студенту:**

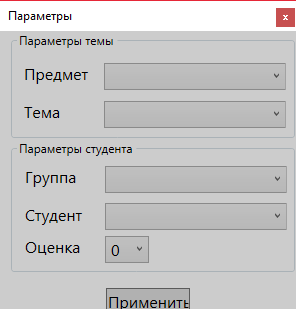


Рисунок 13 – Форма «Добавление темы студенту»

* + 1. **Описание процесса отладки и тестирования приложения**

В процессе написания программы были встречены следующие проблемы и выбраны следующие методы их решения:

* проблема, связанная с кодировкой.
  + 1. **Защита информационной системы**

Защита осуществляется посредством системы ролей, интегрированной в MySQL. Для работы с системой, пользователю необходимо авторизоваться через окно входа, которое открывается при нажатии на кнопку авторизации. Данная система ролей обеспечивает хранение паролей в системных таблицах в зашифрованном виде.

Приложение было защищено от SQL-инъекций с помощью наложения соответствующих масок на поля ввода.

* 1. **Инструкция пользователю** 
     1. **Общие сведения об информационной системе**

Для работы в системе необходимо ее настроить, путем установки необходимых программных средств, и запустить ее сервер. Далее пользователю необходимо авторизоваться в системе через окно «Авторизация», которое откроется при нажатии на кнопку авторизации. Для удачной авторизации необходимо ввести логин (имя роли) и пароль (пароль роли), и нажать кнопку «Войти». В случае если вы еще не зарегистрированы в системе, то вам следует обратиться к администратору, чтобы тот зарегистрировал нового пользователя.

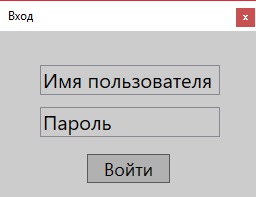


Рисунок 14 – Форма «Аутентификация»

Процедура регистрации состоит в заполнении полей на форме регистрации. Администратору необходимо ввести логин, пароль, роль, имя, фамилию и пол в соответствующих полях. Нажать кнопку «Добавить» для завершения процедуры регистрации, далее вы вернетесь на форму «Авторизация», где вы можете авторизоваться.

После авторизации вы будете возвращены в главное окно программы. Для просмотра всех таблиц следует нажать кнопку «Таблицы». В меню слева есть пункты: «Домой», «Таблицы», «Пользователи», «Настройки», «Отчёт».

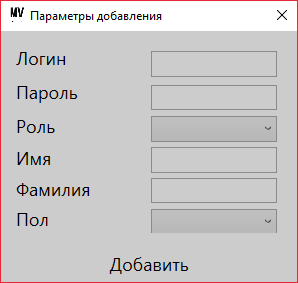


Рисунок 15 – Форма «Регистрация»

Для занятия темы студенту следует нажать кнопку «Таблицы» слева. Далее открыть выпадающее окно и выбрать пункт «Занятые темы».

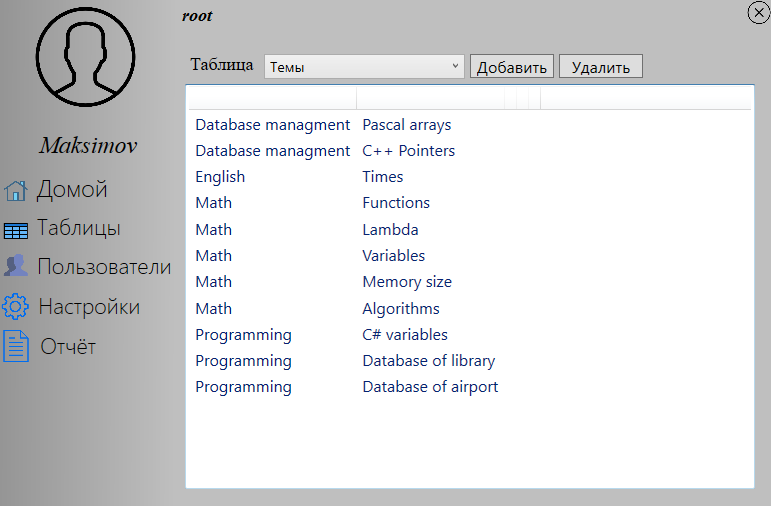


Рисунок 16 – Форма «Таблицы – Темы»

После чего откроется окно «Занятые темы» в котором пользователю следует выбрать либо автоматическое распределение тем, либо ручное. Если выбрать авто-распределение, то следует нажать кнопку «Распределить». В открывшемся окне выбрать группу и предмет, после чего подтвердить действие кнопкой «Принять».

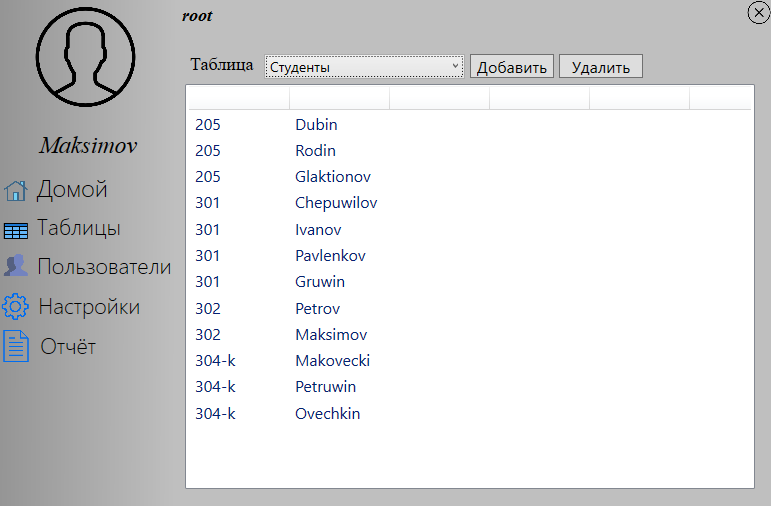


Рисунок 17 – Форма «Таблицы – студенты»

* + 1. **Требования к техническим средствам**

Предъявляются следующие требования к аппаратной части:

* intel core 2 duo или AMD аналог,
* периферия для манипулирования данными,
* монитор,
* подключение к сети интернет на скорости 1 Мбит/с (если информационная система находится на удаленном компьютере).
  + 1. **Требования к программным средствам**

Предъявляются следующие требования к программным средствам:

* сервер MySQL, настроенный на удаленное подключение и содержащий в себе схему информационной системы,
* в операционную систему должен быть установлен набор динамических библиотек, который поставляется вместе с информационной системой.
  + 1. **Настройка информационной системы**

Настройка осуществляется квалифицированным специалистом в области администрирования баз данных. Для успешной работы с базой, ему необходимо будет перенести схему информационной системы в базу MySQL с помощью скриптов, прилагаемых к системе. Он должен воспроизвести данные запросы в пустой базе данных MySQL. После чего необходимо будет скопировать приложение с диска и запустить исполняемый файл «qoorsach.exe». Выполнить вход в систему под пользователем «root». Если вход успешно выполнен, значит настройка проведена верно.

# **ЗАКЛЮЧЕНИЕ**

Курсовой проект посвящен разработке информационной системы, которая позволяет в автоматическом порядке распределить все темы между студентами в группах.

В процессе выполнения курсового проекта:

1. ) изучена предметная область;
2. ) спроектирована и разработана база данных;
3. ) разработано приложение средствами IDE Visual Studio в среде C#/WPF;
4. ) осуществлена защита базы данных посредством использования системы ролей в СУБД MySQL;
5. ) освоены методы моделирования, разработки, проектирования сложной информационной системы, содержащей базу данных и графическое приложение с сетевым взаимодействием.

Система позволяет конечному пользователю вести учёт сдачи работ студентами колледжа.

По итогу в базе данных были созданы следующие объекты:

* 7 таблиц,
* 0 представлений,
* 30 хранимых процедур.

В клиентской части было создано:

* 3 пользовательских класса,
* 3 пользовательских формы,
* Написано более 3000 строк кода,
* Решено несколько локальных задач.

На разработку в конечном счете было затрачено порядка 120 часов.

Дальнейшее развитие и модификация приложения предполагает добавление возможности работы с русской кодировкой.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Dot Net [Электронный ресурс] – URL: https://docs.microsoft.com/ru-ru/dotnet/ (дата обращения: 20.06.2018);
2. MySQL Documentation [Электронный ресурс] – URL: https://dev.mysql.com/doc/ (дата обращения: 20.06.2018).

# **ПРИЛОЖЕНИЕ А SQL скрипты на создание и заполнение базы данных, на создание учетных записей пользователей, на назначение привилегий**

-- MySQL dump 10.13 Distrib 5.5.23, for Win64 (x86)

--

-- Host: localhost Database: mydb

-- ------------------------------------------------------

-- Server version 5.5.23-log

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `edu\_plan`

--

DROP TABLE IF EXISTS `edu\_plan`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `edu\_plan` (

`semester\_number` int(11) NOT NULL,

`hours` int(11) NOT NULL,

`year` int(11) NOT NULL,

`Specs\_id` int(11) NOT NULL,

`Subjects\_id` int(11) NOT NULL,

PRIMARY KEY (`semester\_number`,`year`,`Specs\_id`,`Subjects\_id`),

KEY `fk\_Edu\_plan\_Specs1\_idx` (`Specs\_id`),

KEY `fk\_Edu\_plan\_Subjects1\_idx` (`Subjects\_id`),

CONSTRAINT `fk\_Edu\_plan\_Specs1` FOREIGN KEY (`Specs\_id`) REFERENCES `specs` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `fk\_Edu\_plan\_Subjects1` FOREIGN KEY (`Subjects\_id`) REFERENCES `subjects` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `edu\_plan`

--

LOCK TABLES `edu\_plan` WRITE;

/\*!40000 ALTER TABLE `edu\_plan` DISABLE KEYS \*/;

INSERT INTO `edu\_plan` VALUES (1,900,2018,1,1),(1,1200,2018,2,1),(1,900,2018,3,1),(2,990,2018,2,2);

/\*!40000 ALTER TABLE `edu\_plan` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `engaged\_themes`

--

DROP TABLE IF EXISTS `engaged\_themes`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `engaged\_themes` (

`Students\_id` int(11) NOT NULL,

`Students\_Groups\_id` int(11) NOT NULL,

`Themes\_id` int(11) NOT NULL,

`Themes\_Subjects\_id` int(11) NOT NULL,

`Mark` int(11) DEFAULT NULL,

PRIMARY KEY (`Students\_id`,`Students\_Groups\_id`,`Themes\_id`,`Themes\_Subjects\_id`),

KEY `fk\_Engaged\_Themes\_Students1\_idx` (`Students\_id`,`Students\_Groups\_id`),

KEY `fk\_Engaged\_Themes\_Themes1\_idx` (`Themes\_id`,`Themes\_Subjects\_id`),

CONSTRAINT `fk\_Engaged\_Themes\_Students1` FOREIGN KEY (`Students\_id`, `Students\_Groups\_id`) REFERENCES `students` (`id`, `Groups\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION,

CONSTRAINT `fk\_Engaged\_Themes\_Themes1` FOREIGN KEY (`Themes\_id`, `Themes\_Subjects\_id`) REFERENCES `themes` (`id`, `Subjects\_id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `engaged\_themes`

--

LOCK TABLES `engaged\_themes` WRITE;

/\*!40000 ALTER TABLE `engaged\_themes` DISABLE KEYS \*/;

INSERT INTO `engaged\_themes` VALUES (1,1,11,6,0),(1,5,9,6,0),(2,5,10,6,0);

/\*!40000 ALTER TABLE `engaged\_themes` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `groups`

--

DROP TABLE IF EXISTS `groups`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `groups` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`Specs\_spec\_id` int(11) NOT NULL,

PRIMARY KEY (`id`,`Specs\_spec\_id`),

UNIQUE KEY `name\_UNIQUE` (`name`),

KEY `fk\_Groups\_Specs1\_idx` (`Specs\_spec\_id`),

CONSTRAINT `fk\_Groups\_Specs1` FOREIGN KEY (`Specs\_spec\_id`) REFERENCES `specs` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=6 DEFAULT CHARSET=utf8;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `groups`

--

LOCK TABLES `groups` WRITE;

/\*!40000 ALTER TABLE `groups` DISABLE KEYS \*/;

INSERT INTO `groups` VALUES (5,'205',1),(2,'301',3),(4,'302',2),(1,'304-k',1);

/\*!40000 ALTER TABLE `groups` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `specs`

--

DROP TABLE IF EXISTS `specs`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `specs` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`code` varchar(15) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `code\_UNIQUE` (`code`),

UNIQUE KEY `id\_UNIQUE` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `specs`

--

LOCK TABLES `specs` WRITE;

/\*!40000 ALTER TABLE `specs` DISABLE KEYS \*/;

INSERT INTO `specs` VALUES (1,'Programming in Computer Systems','09.02.03'),(2,'Computer Network','09.02.02'),(3,'Marketing','42.02.01');

/\*!40000 ALTER TABLE `specs` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `students`

--

DROP TABLE IF EXISTS `students`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `students` (

`id` int(11) NOT NULL,

`surname` varchar(45) NOT NULL,

`Groups\_id` int(11) NOT NULL,

PRIMARY KEY (`id`,`Groups\_id`),

KEY `fk\_Students\_Groups1\_idx` (`Groups\_id`),

CONSTRAINT `fk\_Students\_Groups1` FOREIGN KEY (`Groups\_id`) REFERENCES `groups` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `students`

--

LOCK TABLES `students` WRITE;

/\*!40000 ALTER TABLE `students` DISABLE KEYS \*/;

INSERT INTO `students` VALUES (1,'Ovechkin',1),(1,'Ivanov',2),(1,'Maksimov',4),(1,'Rodin',5),(2,'Petruwin',1),(2,'Gruwin',2),(2,'Petrov',4),(2,'Glaktionov',5),(3,'Makovecki',1),(3,'Pavlenkov',2),(3,'Dubin',5),(4,'Chepuwilov',2);

/\*!40000 ALTER TABLE `students` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `subjects`

--

DROP TABLE IF EXISTS `subjects`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `subjects` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `id\_UNIQUE` (`id`),

UNIQUE KEY `name\_UNIQUE` (`name`)

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `subjects`

--

LOCK TABLES `subjects` WRITE;

/\*!40000 ALTER TABLE `subjects` DISABLE KEYS \*/;

INSERT INTO `subjects` VALUES (2,'Database managment'),(5,'English'),(6,'Math'),(1,'Programming');

/\*!40000 ALTER TABLE `subjects` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `themes`

--

DROP TABLE IF EXISTS `themes`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `themes` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` varchar(45) NOT NULL,

`Subjects\_id` int(11) NOT NULL,

PRIMARY KEY (`id`,`Subjects\_id`),

KEY `fk\_Themes\_Subjects1\_idx` (`Subjects\_id`),

CONSTRAINT `fk\_Themes\_Subjects1` FOREIGN KEY (`Subjects\_id`) REFERENCES `subjects` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB AUTO\_INCREMENT=13 DEFAULT CHARSET=utf8;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `themes`

--

LOCK TABLES `themes` WRITE;

/\*!40000 ALTER TABLE `themes` DISABLE KEYS \*/;

INSERT INTO `themes` VALUES (1,'Database of airport',1),(1,'Pascal arrays',2),(2,'Database of library',1),(2,'C++ Pointers',2),(6,'C# variables',1),(7,'Algorithms',6),(8,'Memory size',6),(9,'Variables',6),(10,'Lambda',6),(11,'Functions',6),(12,'Times',5);

/\*!40000 ALTER TABLE `themes` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `users`

--

DROP TABLE IF EXISTS `users`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `users` (

`login` varchar(45) CHARACTER SET utf8 NOT NULL,

`role` varchar(45) CHARACTER SET utf8 NOT NULL,

`name` varchar(45) CHARACTER SET utf8 NOT NULL,

`surname` varchar(45) CHARACTER SET utf8 NOT NULL,

`gender` varchar(45) CHARACTER SET utf8 DEFAULT NULL,

UNIQUE KEY `login\_UNIQUE` (`login`)

) ENGINE=InnoDB DEFAULT CHARSET=utf16;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `users`

--

LOCK TABLES `users` WRITE;

/\*!40000 ALTER TABLE `users` DISABLE KEYS \*/;

INSERT INTO `users` VALUES ('depot','Depot','Aleksey','Panin','Male'),('larionova','Teacher','Elena','Larionova','Female'),('login','Teacher','Masha','Ivanova','Female'),('maksimov','Student','Ilya','Maksimov','Male'),('root','root','Ilya','Maksimov','');

/\*!40000 ALTER TABLE `users` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2018-06-20 23:10:27

# **ПРИЛОЖЕНИЕ Б Код программы**

**Файл MainWindow.xaml.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using MySql.Data;

using MySql.Data.MySqlClient;

using System.Diagnostics;

using System.IO;

using PdfSharp;

using PdfSharp.Drawing;

using PdfSharp.Pdf;

using PdfSharp.Pdf.IO;

namespace MyApp

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

//private const string DEBUG\_CONNSTR = "server=localhost;persistsecurityinfo=True;database=mydb;allowuservariables=True;SslMode=none;user id = root; password = root";

MyApp.Window1 win = new MyApp.Window1();

addForm form = new addForm();

TablesAddForm tablesAddForm;

XFont titleFont = new XFont("Impact", 25);

XFont font = new XFont("Times New Roman", 10);

//XBrush brush = new XBrush();

public class Data

{

public int id;

public string login;

}

private class S\_S

{

public string student;

public string theme;

}

private class BigData

{

public int student\_id { get; set; }

public string student { get; set; }

public int theme\_id { get; set; }

public string theme { get; set; }

}

List<Data> idLogin = new List<Data>();

public MainWindow()

{

InitializeComponent();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

}

private void Grid\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

this.DragMove();

}

private void Image\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

Environment.Exit(0);

}

private void Image\_MouseEnter(object sender, MouseEventArgs e)

{

this.closeButton.Height = this.closeButton.Height - 2;

}

private void Image\_MouseLeave(object sender, MouseEventArgs e)

{

this.closeButton.Height = this.closeButton.Height + 2;

}

public class MyItem2

{

public string Col1 { get; set; }

public string Col2 { get; set; }

public string Col3 { get; set; }

public string Col4 { get; set; }

public string Col5 { get; set; }

}

private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

distribute.Visibility = Visibility.Hidden;

if ( common.nRole == "Admin" || common.nRole == "root" ||

common.nRole == "Teacher" || common.nRole == "Depot" )

{

ButtonAdd.IsEnabled = true;

ButtonDelete.IsEnabled = true;

distribute.IsEnabled = true;

}

//Console.WriteLine(comboBoxTable.SelectedIndex);

string req = "";

switch(comboBoxTable.SelectedIndex)

{

case 0:

req = "edu\_plan"; break;

case 1:

req = "engaged\_themes";

distribute.Visibility = Visibility.Visible;

break;

case 2:

req = "groups"; break;

case 3:

req = "specs"; break;

case 4:

req = "students"; break;

case 5:

req = "subjects"; break;

case 6:

req = "themes"; break;

}

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

//Console.WriteLine(connStr);

MySqlConnection conn = new MySqlConnection(connStr);

try

{

Console.WriteLine("Попытка подключения...");

conn.Open();

string sql = "call show\_" + req;

//Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

listViewTables.Items.Clear();

string s1 = "", s2 = "", s3 = "", s4 = "", s5 = "";

while (rdr.Read())

{

try

{

s1 = rdr[0].ToString();

s2 = rdr[1].ToString();

s3 = rdr[2].ToString();

s4 = rdr[3].ToString();

s5 = rdr[4].ToString();

}

catch(Exception ex)

{

//Console.WriteLine(ex.ToString());

}

listViewTables.Items.Add(new MyItem2

{

Col1 = s1, Col2 = s2,

Col3 = s3, Col4 = s4,

Col5 = s5

});

}

rdr.Close();

conn.Close();

}

catch (MySqlException ex)

{

//Console.WriteLine(ex.ToString());

}

}

private void Image\_MouseEnter\_1(object sender, MouseEventArgs e)

{

this.Avatar.Height = this.Avatar.Height - 5;

}

private void Image\_MouseLeave\_1(object sender, MouseEventArgs e)

{

this.Avatar.Height = this.Avatar.Height + 5;

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

Console.WriteLine("Tables\_click");

gHome.Visibility = Visibility.Hidden;

gUser.Visibility = Visibility.Hidden;

gUsers.Visibility = Visibility.Hidden;

gSettings.Visibility = Visibility.Hidden;

gTables.Visibility = Visibility.Visible;

}

private void Home\_Click(object sender, RoutedEventArgs e)

{

Console.WriteLine("Home\_click");

this.gUser.Visibility = Visibility.Hidden;

this.gUsers.Visibility = Visibility.Hidden;

this.gSettings.Visibility = Visibility.Hidden;

this.gTables.Visibility = Visibility.Hidden;

this.gHome.Visibility = Visibility.Visible;

}

private void Users\_Click(object sender, RoutedEventArgs e)

{

Console.WriteLine("User\_click");

this.gHome.Visibility = Visibility.Hidden;

this.gUser.Visibility = Visibility.Hidden;

this.gSettings.Visibility = Visibility.Hidden;

this.gTables.Visibility = Visibility.Hidden;

this.gUsers.Visibility = Visibility.Visible;

}

private void Settings\_Click(object sender, RoutedEventArgs e)

{

Console.WriteLine("Settings\_click");

this.gHome.Visibility = Visibility.Hidden;

this.gUser.Visibility = Visibility.Hidden;

this.gUsers.Visibility = Visibility.Hidden;

this.gTables.Visibility = Visibility.Hidden;

this.gSettings.Visibility = Visibility.Visible;

}

private void Avatar\_MouseLeftButtonDown\_1(object sender, MouseButtonEventArgs e)

{

Console.WriteLine("User\_click");

this.gHome.Visibility = Visibility.Hidden;

this.gUsers.Visibility = Visibility.Hidden;

this.gTables.Visibility = Visibility.Hidden;

this.gSettings.Visibility = Visibility.Hidden;

this.gUser.Visibility = Visibility.Visible;

}

private void Home\_Copy\_Click(object sender, RoutedEventArgs e) // Авторизация

{

win = new MyApp.Window1();

win.Show();

}

public class MyItem

{

public string Login { get; set; }

public string Role { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Gender { get; set; }

}

private void setParams(string connStr)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

//Console.WriteLine("Попытка подключения...");

conn.Open();

string sql = "getmyinfo('" + common.username + "')";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Read();

string name = "Console - No Answer";

string surname = "Console - No Answer";

string role = "Console - No Answer";

string gender = "Console - No Answer";

try

{

name = rdr[0].ToString();

surname = rdr[1].ToString();

role = rdr[2].ToString();

gender = rdr[3].ToString();

}

catch (Exception)

{

Console.WriteLine("Пусто");

}

rdr.Close();

//Console.WriteLine(name);

Name\_Copy.Content = name;

Surname\_Copy.Content = surname;

Surname\_Copy1.Content = role;

Home\_Copy.Content = surname;

rights.Content = role;

common.nRole = role;

if (gender == "Male")

try

{

Avatar.Source = new BitmapImage(

new Uri("pack://application:,,,/MyApp;component/icons/male.png"));

Avatar\_Copy.Source = new BitmapImage(

new Uri("pack://application:,,,/MyApp;component/icons/male.png"));

}

catch (Exception ex)

{

//Console.WriteLine(ex.ToString());

//Console.WriteLine(Avatar.Source.ToString());

}

else if (gender == "Female")

try

{

Avatar.Source = new BitmapImage(

new Uri("pack://application:,,,/MyApp;component/icons/female.png"));

Avatar\_Copy.Source = new BitmapImage(

new Uri("pack://application:,,,/MyApp;component/icons/female.png"));

}

catch (Exception ex)

{

//Console.WriteLine(ex.ToString());

//Console.WriteLine(Avatar.Source.ToString());

}

else try

{

Avatar.Source = new BitmapImage(

new Uri("pack://application:,,,/MyApp;component/icons/user.png"));

Avatar\_Copy.Source = new BitmapImage(

new Uri("pack://application:,,,/MyApp;component/icons/user.png"));

}

catch (Exception ex)

{

// Console.WriteLine(ex.ToString());

// Console.WriteLine(Avatar.Source.ToString());

}

if (common.nRole == "Admin" || common.nRole == "root")

{

ButtonAdd2.IsEnabled = true;

ButtonDelete2.IsEnabled = true;

sql = "call show\_users";

cmd = new MySqlCommand(sql, conn);

rdr = cmd.ExecuteReader();

listViewUsers.Items.Clear();

listViewTables.Items.Clear();

int counter = 0;

idLogin = new List<Data>();

while (rdr.Read())

try

{

listViewUsers.Items.Add(new MyItem

{

Login = rdr[0].ToString(),

Role = rdr[1].ToString(),

Name = rdr[2].ToString(),

Surname = rdr[3].ToString(),

Gender = rdr[4].ToString()

});

Data it = new Data();

it.id = counter;

it.login = rdr[0].ToString();

idLogin.Add(it);

counter++;

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

rdr.Close();

conn.Close();

} else {

ButtonAdd.IsEnabled = false;

ButtonDelete.IsEnabled = false;

ButtonAdd2.IsEnabled = false;

ButtonDelete2.IsEnabled = false;

listViewUsers.Items.Clear();

}

} catch (Exception ex)

{

// Console.WriteLine(ex.ToString());

}

}

private void setUsersTable(string connStr)

{

// Console.WriteLine("wannaToCreate" + common.username + "\t" + common.password);

MySqlConnection conn = new MySqlConnection(connStr);

try

{

//Console.WriteLine("Попытка подключения...");

conn.Open();

string sql = "call add\_User(@username,@password,@role,@name,@surname,@gender)";

MySqlCommand cmd = new MySqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@username", common.nUsername);

cmd.Parameters.AddWithValue("@password", common.nPassword);

cmd.Parameters.AddWithValue("@role", common.nRole);

cmd.Parameters.AddWithValue("@name", common.nName);

cmd.Parameters.AddWithValue("@surname", common.nSurname);

cmd.Parameters.AddWithValue("@gender", common.nGender);

cmd.ExecuteNonQuery();

conn.Close();

Data it = new Data();

it.id = idLogin.Count;

it.login = common.nUsername;

idLogin.Add(it);

}

catch (MySqlException ex)

{

MessageBox.Show("Ошибка добавления");

Console.WriteLine(ex.ToString());

}

}

private void addNewTheme(string connStr)

{

}

private void Window\_GotFocus(object sender, EventArgs e) // Обработчик

{

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

if (common.wannaToConnect)

{

common.wannaToConnect = false;

setParams(connStr);

}

else if (common.wannaToCreateUser)

{

common.wannaToCreateUser = false;

setUsersTable(connStr);

setParams(connStr);

}

else if (common.wannaToCreateTheme)

{

common.wannaToCreateTheme = false;

addNewTheme(common.connectionString);

}

}

private void Button\_Click\_2(object sender, RoutedEventArgs e) // Добавить кнопка

{

form = new addForm();

form.Show();

}

private void Button\_Click\_3(object sender, RoutedEventArgs e) // Удалить кнопка

{

try

{

bool deleted = false;

int id = listViewUsers.SelectedIndex;

foreach (Data it in idLogin)

if (it.id == id && !deleted)

if (MessageBox.Show(

"Уверены что хотите удалить пользователя " + it.login + "?",

"Подтвердите ",

MessageBoxButton.YesNo) == MessageBoxResult.Yes)

{

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

Console.WriteLine("Попытка подключения...");

conn.Open();

string sql = "call rm\_user(@user)";

MySqlCommand cmd = new MySqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@user",it.login);

cmd.ExecuteNonQuery();

conn.Close();

MessageBox.Show("Пользователь " + it.login + " успешно удалён!");

listViewUsers.Items.RemoveAt(it.id);

deleted = true;

//idLogin.Remove(it);

setParams(connStr);

}

catch (MySqlException ex)

{

Console.WriteLine(ex.ToString());

}

}

else return;

}

catch(Exception ex)

{

// MessageBox.Show("Выберите пользователя для удаления","Ошибка");

Console.WriteLine(ex.ToString());

}

}

private void MainGrid\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

this.DragMove();

}

private void ButtonAdd\_Click(object sender, RoutedEventArgs e)

{

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

if (comboBoxTable.SelectedValue != null)

if (comboBoxTable.Text == "Занятые темы")

{

tablesAddForm = new TablesAddForm(connStr, "add");

tablesAddForm.Show();

} else

{

AdaptiveWindow AW = new AdaptiveWindow(connStr, comboBoxTable.Text, "Добавить");

AW.Show();

}

}

private void ButtonDelete\_Click(object sender, RoutedEventArgs e)

{

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

if (comboBoxTable.SelectedValue != null)

if (comboBoxTable.Text == "Занятые темы")

{

tablesAddForm = new TablesAddForm(connStr, "delete");

tablesAddForm.Show();

} else

{

AdaptiveWindow AW = new AdaptiveWindow(connStr, comboBoxTable.Text, "Удалить");

AW.Show();

}

}

private void comboBoxRole\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

private void distribute\_Click(object sender, RoutedEventArgs e)

{

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

AdaptiveWindow AW = new AdaptiveWindow(connStr, "Распределить", "Применить");

AW.Show();

}

private void drawLines(XGraphics gfx, PdfPage page, int w1 = -1, int w2 = -1, int w3 = -1, int w4 = -1, int w5 = -1)

{

gfx.DrawLine(XPens.Black, w1, 50, w1, page.Height - 55);

gfx.DrawLine(XPens.Black, w2, 50, w2, page.Height - 55);

gfx.DrawLine(XPens.Black, w3, 50, w3, page.Height - 55);

gfx.DrawLine(XPens.Black, w4, 50, w4, page.Height - 55);

gfx.DrawLine(XPens.Black, w5, 50, w5, page.Height - 55);

}

private void drawTable(XGraphics gfx, PdfPage page)

{

gfx.DrawRectangle(XPens.Black, new XRect(25, 50, page.Width - 50, page.Height - 105));

for (int i = 50; i < page.Height - 70; i += 15)

gfx.DrawLine(XPens.Black, 25, i, page.Width - 25, i);

}

private void drawThemes(PdfPage page) // Темы

{

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

gfx.DrawString("Themes", titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

drawLines(gfx, page, 300);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_Themes()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

int y = 60, x = 26;

while( rdr.Read() )

{

gfx.DrawString(rdr[0].ToString(), font, XBrushes.Black,

new XRect(x, y, 250, 0));

gfx.DrawString(rdr[1].ToString(), font, XBrushes.Black,

new XRect(x + 276, y, 250, 0));

y += 15;

}

}

finally

{

conn.Close();

}

}

private void drawEducationPlan(PdfPage page) // Учебный план

{

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

gfx.DrawString("Education Plan", titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

drawLines(gfx, page, 50,70,100,350);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_edu\_plan()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

int y = 60, x = 26;

while (rdr.Read())

{

gfx.DrawString(rdr[0].ToString(), font, XBrushes.Black,

new XRect(x, y, 250, 0));

gfx.DrawString(rdr[1].ToString(), font, XBrushes.Black,

new XRect(x + 27, y, 250, 0));

gfx.DrawString(rdr[2].ToString(), font, XBrushes.Black,

new XRect(x + 46, y, 250, 0));

gfx.DrawString(rdr[3].ToString(), font, XBrushes.Black,

new XRect(x + 75, y, 250, 0));

gfx.DrawString(rdr[4].ToString(), font, XBrushes.Black,

new XRect(x + 326, y, 250, 0));

y += 15;

}

}

finally

{

conn.Close();

}

}

private void drawEngagedThemes(PdfPage page) // Занятые темы

{

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

gfx.DrawString("Engaged Themes", titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

drawLines(gfx, page, 60,150,300,550);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_engaged\_themes()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

int y = 60, x = 26;

while (rdr.Read())

{

gfx.DrawString(rdr[0].ToString(), font, XBrushes.Black,

new XRect(x, y, 250, 0));

gfx.DrawString(rdr[1].ToString(), font, XBrushes.Black,

new XRect(x + 36, y, 250, 0));

gfx.DrawString(rdr[2].ToString(), font, XBrushes.Black,

new XRect(x + 125, y, 250, 0));

gfx.DrawString(rdr[3].ToString(), font, XBrushes.Black,

new XRect(x + 275, y, 250, 0));

gfx.DrawString(rdr[4].ToString(), font, XBrushes.Black,

new XRect(x + 530, y, 250, 0));

y += 15;

}

}

finally

{

conn.Close();

}

}

private void DrawEngagedThemesInGroup(PdfPage page, PdfDocument document,int group\_id)

{

int y = 60, x = 26;

page = document.AddPage();

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

string name,sql;

MySqlCommand cmd;

MySqlDataReader rdr;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

sql = "SELECT name FROM groups WHERE id = @myid";

cmd = new MySqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@myid",group\_id);

rdr = cmd.ExecuteReader();

rdr.Read();

name = rdr[0].ToString();

}

finally

{

conn.Close();

}

gfx.DrawString("Group " + name, titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

List<Data> list2 = new List<Data>();

conn = new MySqlConnection(connStr);

try

{

conn.Open();

sql = "SELECT id,name FROM subjects ORDER BY id";

cmd = new MySqlCommand(sql, conn);

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.login = rdr[1].ToString();

list2.Add(it);

}

}

finally

{

conn.Close();

}

foreach (Data d in list2)

{

string subject\_name = d.login;

List<S\_S> s\_s = new List<S\_S>();

List<BigData> BDlist = new List<BigData>();

sql = "call autoSplit(@group\_id,@subject\_id)";

conn = new MySqlConnection(connStr);

cmd = new MySqlCommand(sql, conn);

cmd.Parameters.AddWithValue("@group\_id", group\_id);

cmd.Parameters.AddWithValue("@subject\_id", d.id);

conn.Open();

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

string studentTheme = rdr[3].ToString();

string studentName = rdr[1].ToString();

BigData it = new BigData();

it.student\_id = Convert.ToInt32(rdr[0].ToString());

it.student = studentName;

it.theme\_id = Convert.ToInt32(rdr[2].ToString());

it.theme = studentTheme;

BDlist.Add(it);

}

conn.Close();

for (int n = 0; n < BDlist.Count - 1; ++n)

{

BDlist.RemoveAll(

it => s\_s.Exists(

it2 => ((it2.student == it.student) || (it2.theme == it.theme))

)

);

int std\_id = 0, thm\_id = 0;

string std = "", thm = "";

try

{

std\_id = BDlist[0].student\_id;

std = BDlist[0].student;

thm\_id = BDlist[0].theme\_id;

thm = BDlist[0].theme;

}

catch (Exception ex)

{

//Console.WriteLine(ex.ToString());

}

S\_S s = new S\_S();

s.student = std;

s.theme = thm;

s\_s.Add(s);

}

foreach (S\_S s in s\_s) Console.WriteLine(s.student + " " + s.theme);

//Console.WriteLine(d.login);

foreach (BigData bd in BDlist)

{

foreach (S\_S s in s\_s)

{

/\*gfx.DrawString(s.student, font, XBrushes.Black,

new XRect(x + 100, y, 250, 0));

gfx.DrawString(s.theme, font, XBrushes.Black,

new XRect(x + 200, y, 250, 0));\*/

/\*

gfx.DrawString(bd.student, font, XBrushes.Black,

new XRect(x + 500, y, 250, 0));

gfx.DrawString(bd.theme, font, XBrushes.Black,

new XRect(x + 400, y, 250, 0));\*/

//Console.WriteLine(d.login + " ---- " + s.student + " ---- " + s.theme);

}

//Console.WriteLine(name + "\t-\t" + bd.student + "\t-\t" + d.login + "\t-\t" + bd.theme);

//gfx.DrawString(d.login, font, XBrushes.Black,

// new XRect(x, y, 300, 0));

y += 15;

}

//Console.WriteLine("-------------------------\n");

}

drawLines(gfx, page, 180,400);

}

private void DrawFreeThemes(PdfPage page,PdfDocument document) // Larionova's last hope

{

List<Data> list1 = new List<Data>();

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT id, name FROM groups ORDER BY id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.login = rdr[1].ToString();

list1.Add(it);

}

}

finally

{

conn.Close();

}

foreach (Data d in list1) DrawEngagedThemesInGroup(page,document,d.id);

}

private void drawGroups(PdfPage page) // Группы

{

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

gfx.DrawString("Groups", titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

drawLines(gfx, page, 60);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_groups()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

int y = 60, x = 26;

while (rdr.Read())

{

gfx.DrawString(rdr[0].ToString(), font, XBrushes.Black,

new XRect(x, y, 250, 0));

gfx.DrawString(rdr[1].ToString(), font, XBrushes.Black,

new XRect(x + 36, y, 250, 0));

y += 15;

}

}

finally

{

conn.Close();

}

}

private void drawStudents(PdfPage page) // Студенты

{

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

gfx.DrawString("Students", titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

drawLines(gfx, page, 60);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_students()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

int y = 60, x = 26;

while (rdr.Read())

{

gfx.DrawString(rdr[0].ToString(), font, XBrushes.Black,

new XRect(x, y, 250, 0));

gfx.DrawString(rdr[1].ToString(), font, XBrushes.Black,

new XRect(x + 36, y, 250, 0));

y += 15;

}

}

finally

{

conn.Close();

}

}

private void drawSubjects(PdfPage page) // Предметы

{

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

gfx.DrawString("Subjects", titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

//drawLines(gfx, page, 50);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_subjects()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

int y = 60, x = 26;

while (rdr.Read())

{

gfx.DrawString(rdr[0].ToString(), font, XBrushes.Black,

new XRect(x, y, 250, 0));

y += 15;

}

}

finally

{

conn.Close();

}

}

private void drawSpecs(PdfPage page) // Специальности

{

XGraphics gfx = XGraphics.FromPdfPage(page);

drawTable(gfx, page);

gfx.DrawString("Specialties", titleFont, XBrushes.Black,

new XRect(50, 10, page.Width, 100),

XStringFormats.TopLeft);

drawLines(gfx, page, 420);

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_specs()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

int y = 60, x = 26;

while (rdr.Read())

{

gfx.DrawString(rdr[0].ToString(), font, XBrushes.Black,

new XRect(x, y, 250, 0));

gfx.DrawString(rdr[1].ToString(), font, XBrushes.Black,

new XRect(x + 400, y, 250, 0));

y += 15;

}

}

finally

{

conn.Close();

}

}

private void Report\_Click(object sender, RoutedEventArgs e)

{

bool error = false;

string connStr = connectionString.Text +

"user id=" + common.username +

";password=" + common.password;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

conn.Close();

}

catch(MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка подключения.\nВозможно вы не авторизованы.");

}

if (!error)

{

// Create a new PDF document

PdfDocument document = new PdfDocument();

document.Info.Title = "Report";

// Create an empty page

PdfPage page = document.AddPage();

drawThemes(page);

page = document.AddPage();

drawEducationPlan(page);

page = document.AddPage();

drawEngagedThemes(page);

page = document.AddPage();

drawGroups(page);

page = document.AddPage();

drawStudents(page);

page = document.AddPage();

drawSubjects(page);

page = document.AddPage();

drawSpecs(page);

//page = document.AddPage();

DrawFreeThemes(page,document);

// Save the document...

const string filename = "Report.pdf";

document.Save(filename);

// ...and start a viewer.

Process.Start(filename);

}

}

}

}

**Файл AdaptiveWindow.xaml.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using MySql.Data;

using MySql.Data.MySqlClient;

namespace MyApp

{

public partial class AdaptiveWindow : Window

{

public class Data

{

public int id { get; set; }

public string name { get; set; }

}

List<Data> list = new List<Data>();

List<Data> list2 = new List<Data>();

public bool done;

public string connStr;

public string type;

public string action;

public AdaptiveWindow(string connStr, string type, string action)

{

InitializeComponent();

done = false;

Title = type;

this.connStr = connStr;

this.type = type;

this.action = action;

ActionButton.Content = action;

if (type == "Распределить") gAuto.Visibility = Visibility.Visible;

if (type == "Темы" && action == "Добавить") gThemesAdd.Visibility = Visibility.Visible;

if (type == "Темы" && action == "Удалить") gThemesDelete.Visibility = Visibility.Visible;

if (type == "Специальности" && action == "Добавить") gSpecAdd.Visibility = Visibility.Visible;

if (type == "Специальности" && action == "Удалить") gSpecDelete.Visibility = Visibility.Visible;

if (type == "Студенты" && action == "Добавить") gStudentAdd.Visibility = Visibility.Visible;

if (type == "Студенты" && action == "Удалить") gStudentDelete.Visibility = Visibility.Visible;

if (type == "Группы" && action == "Добавить") gGroupAdd.Visibility = Visibility.Visible;

if (type == "Группы" && action == "Удалить") gGroupDelete.Visibility = Visibility.Visible;

if (type == "Предметы" && action == "Добавить") gSubjectAdd.Visibility = Visibility.Visible;

if (type == "Предметы" && action == "Удалить") gSubjectDelete.Visibility = Visibility.Visible;

if (type == "Учебный план" && action == "Добавить") gEducationPlanAdd.Visibility = Visibility.Visible;

if (type == "Учебный план" && action == "Удалить") gEducationPlanDelete.Visibility = Visibility.Visible;

if (type == "Распределить") DistributeThemes();

if (type == "Темы" && action == "Добавить") ThemesAdd();

if (type == "Темы" && action == "Удалить") ThemesDelete();

if (type == "Специальности" && action == "Добавить") SpecsAdd();

if (type == "Специальности" && action == "Удалить") SpecsDelete();

if (type == "Студенты" && action == "Добавить") StudentsAdd();

if (type == "Студенты" && action == "Удалить") StudentsDelete();

if (type == "Группы" && action == "Добавить") GroupsAdd();

if (type == "Группы" && action == "Удалить") GroupsDelete();

if (type == "Предметы" && action == "Добавить") SubjectsAdd();

if (type == "Предметы" && action == "Удалить") SubjectsDelete();

if (type == "Учебный план" && action == "Добавить") EduPlanAdd();

if (type == "Учебный план" && action == "Удалить") EduPlanDelete();

}

public bool checkString(string str)

{

foreach (char c in str)

if (!System.Text.RegularExpressions.Regex.IsMatch(c.ToString(), "^[a-zA-Z]"))

{

Console.Write(c);

return false;

}

return true;

}

public void ThemesAdd()

{

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM subjects ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

try

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

}

rdr.Close();

}

finally

{

conn.Close();

}

foreach (Data d in list)

gThemesAddSubject.Items.Add(d.name);

}

else

{

bool error = false;

Console.WriteLine("checked string");

MySqlConnection conn = new MySqlConnection(connStr);

try

{

int id = 0;

foreach (Data d in list)

if (d.name == gThemesAddSubject.Text)

id = d.id;

conn.Open();

string sql = "call add\_themes('" + gThemesAddName.Text.ToString() +

"','" + id + "')";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteReader();

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка добавления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error) MessageBox.Show("Тема: " + gThemesAddName.Text + "\nуспешно добавлена!");

}

}

public void ThemesDelete()

{

// List<object> list = new List<object>();

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM subjects ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

gThemesDeleteSubject.Items.Clear();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

/\*list.Add(new {

id =Convert.ToInt32(rdr[0].ToString()),

name =rdr[1].ToString()

}); \*/

Console.WriteLine(it.id + it.name);

gThemesDeleteSubject.Items.Add(it.name);

}

catch (Exception ex)

{

MessageBox.Show(

"Произошла ошибка\n"

+ ex.ToString());

}

list.Add(it);

}

rdr.Close();

}

finally

{

conn.Close();

}

}

else

{

bool error = false;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

int id = 0;

int idSubject = 0;

foreach (Data d in list)

if (d.name == gThemesDeleteSubject.Text)

id = d.id;

foreach (Data d in list2)

if (d.name == gThemesDeleteName.Text)

idSubject = d.id;

conn.Open();

string sql = "call rm\_themes('" + idSubject +

"','" + id + "')";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteReader();

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка удаления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error)

MessageBox.Show("Тема: " +

gThemesAddName.Text +

"\nуспешно удалена!");

}

}

public void SpecsAdd()

{

if (done)

{

bool error = false;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

string sql = "call add\_specs('" +

gSpecAddName.Text + "','" +

gSpecAddCode.Text + "')";

Console.WriteLine(sql);

conn.Open();

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteNonQuery();

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка добавления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error) MessageBox.Show("Специальность " + gSpecAddName.Text + " успешно добавлена!");

}

}

public void SpecsDelete()

{ // gSpecDeleteName

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT id, name FROM specs ORDER BY id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

try

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

}

rdr.Close();

}

finally

{

conn.Close();

}

foreach (Data d in list)

gSpecDeleteName.Items.Add(d.name);

}

else

{

bool error = false;

int id = 0;

foreach (Data d in list)

if (d.name == gSpecDeleteName.Text)

id = d.id;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

string sql = "call rm\_specs('" + id + "')";

Console.WriteLine(sql);

conn.Open();

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteNonQuery();

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка удаления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error) MessageBox.Show("Специальность " + gSpecDeleteName.Text + " успешно добавлена!");

}

}

public void StudentsAdd()

{

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM groups ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

gStudentAddGroup.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

}

else

{

int lastId = 0;

int id = 0;

bool error = false;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

foreach (Data d in list)

if (d.name == gStudentAddGroup.Text)

id = d.id;

string sql = "SELECT max(id) FROM mydb.students WHERE groups\_id = " + id;

Console.WriteLine(sql);

conn.Open();

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

MySqlDataReader rdr = cmd.ExecuteReader();

rdr.Read();

try

{

lastId = Convert.ToInt32(rdr[0].ToString());

} catch (Exception ex)

{

//Console.WriteLine(ex.ToString());

}

lastId++;

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка добавления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call add\_students('" +

lastId + "','" +

gStudentAddSurname.Text + "','" +

id + "')";

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteNonQuery();

} catch (MySqlException ex)

{

error = true;

MessageBox.Show("Произошла ошибка:\n" + ex.ToString());

}

if (!error) MessageBox.Show("Cтудент " + gStudentAddSurname.Text + " успешно добавлен!");

} finally

{

conn.Close();

}

}

}

public void StudentsDelete()

{

// gStudentDeleteGroup - группа

// gStudentDeleteStudent - студент

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM groups ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

gStudentDeleteGroup.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

}

else

{

bool error = false;

int myId = 0, groupsId = 0;

foreach (Data d in list)

if (d.name == gStudentDeleteGroup.Text)

myId = d.id;

foreach (Data d in list2)

if (d.name == gStudentDeleteStudent.Text)

groupsId = d.id;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql =

"call rm\_students('" +

groupsId + "','" +

myId + "')";

Console.WriteLine(sql);

try

{

MySqlCommand cmd = new MySqlCommand(sql, conn);

cmd.ExecuteNonQuery();

} catch (MySqlException ex)

{

error = true;

MessageBox.Show("Произошла ошибка:\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error) MessageBox.Show(

"Студент " +

gStudentDeleteStudent.Text +

" успешно удалён!");

}

}

public void GroupsAdd()

{

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT id, name FROM specs GROUP BY id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

try

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

//Console.WriteLine(it.id + it.name);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

}

rdr.Close();

}

finally

{

conn.Close();

}

foreach (Data d in list)

{

gGroupAddSpec.Items.Add(d.name);

}

} else {

bool ok = checkString(gGroupAddName.Text.ToString());

bool duplicate = false;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

int id = 0;

foreach (Data d in list)

if (d.name == gGroupAddSpec.Text.ToString())

id = d.id;

conn.Open();

string sql = "call add\_groups(" +

"'" + gGroupAddName.Text.ToString() + "'," +

"'" + id.ToString() + "')";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteNonQuery();

} catch (MySqlException ex)

{

duplicate = true;

Console.WriteLine(ex.ToString());

}

}

finally

{

conn.Close();

}

if (!duplicate)

MessageBox.Show("Группа " + gGroupAddName.Text + " успешно добавлена!");

else MessageBox.Show("Группа " + gGroupAddName.Text + " уже существует");

}

}

public void GroupsDelete()

{

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT id, name FROM groups GROUP BY id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

try

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

//Console.WriteLine(it.id + it.name);

}

catch (Exception ex)

{

//Console.WriteLine(ex.ToString());

}

}

rdr.Close();

}

finally

{

conn.Close();

}

foreach (Data d in list)

{

gGroupDeleteGroup.Items.Add(d.name);

}

} else {

MySqlConnection conn = new MySqlConnection(connStr);

try

{

int id = 0;

foreach (Data d in list)

if (d.name == gGroupDeleteGroup.Text.ToString())

id = d.id;

conn.Open();

string sql = "call rm\_groups('" + id + "')";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteNonQuery();

}

catch (MySqlException ex)

{

//Console.WriteLine(ex.ToString());

}

}

finally

{

conn.Close();

}

MessageBox.Show("Группа " + gGroupDeleteGroup.Text + " удалена");

}

}

public void SubjectsAdd()

{

if (done)

{

bool error = false;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call add\_subjects('" + gSubjectAddName.Text.ToString() + "')";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteReader();

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка добавления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error) MessageBox.Show("Предмет: " + gSubjectAddName.Text + "\nуспешно добавлен!");

}

}

public void SubjectsDelete()

{

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM subjects ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

try

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка\n\n" + ex.ToString());

//Console.WriteLine(ex.ToString());

}

}

rdr.Close();

}

finally

{

conn.Close();

}

foreach (Data d in list)

gSubjectDeleteSubject.Items.Add(d.name);

} else {

MySqlConnection conn = new MySqlConnection(connStr);

try

{

int id = 0;

foreach (Data d in list)

if (d.name == gSubjectDeleteSubject.Text)

id = d.id;

conn.Open();

string sql = "call rm\_subjects('" + id.ToString() + "')";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

cmd.ExecuteReader();

}

finally

{

conn.Close();

}

MessageBox.Show("Предмет " + gSubjectDeleteSubject.Text + " успешно удалён!");

}

}

public void EduPlanAdd()

{

// gEducationPlanAddSpec

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM specs ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

gEducationPlanAddSpec.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM subjects ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list2.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

gEducationPlanAddSubject.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

}

else

{

int specId = 0;

int subjectId = 0;

bool error = false;

MySqlConnection conn = new MySqlConnection(connStr);

try

{ // gEducationPlanAddSubject

foreach (Data d in list)

if (d.name == gEducationPlanAddSpec.Text)

specId = d.id;

foreach (Data d in list2)

if (d.name == gEducationPlanAddSubject.Text)

subjectId = d.id;

string sql = "call add\_edu\_plan('" +

gEducationPlanAddSemester.Text + "','" +

gEducationPlanAddHours.Text + "','" +

gEducationPlanAddYear.Text + "','" +

specId + "','" +

subjectId + "')";

Console.WriteLine(sql);

conn.Open();

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteNonQuery();

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка добавления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error) MessageBox.Show("Запись успешно добавлена!");

}

}

public void EduPlanDelete()

{

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM specs ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

gEducationPlanDeleteSpec.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT Id, name FROM subjects ORDER BY Id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list2.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

gEducationPlanDeleteSubject.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

}

else

{

int specId = 0;

int subjectId = 0;

bool error = false;

MySqlConnection conn = new MySqlConnection(connStr);

try

{ // gEducationPlanAddSubject

foreach (Data d in list)

if (d.name == gEducationPlanDeleteSpec.Text)

specId = d.id;

foreach (Data d in list2)

if (d.name == gEducationPlanDeleteSubject.Text)

subjectId = d.id;

string sql = "call rm\_edu\_plan('" +

gEducationPlanDeleteSemester.Text + "','" +

//gEducationPlanDeleteHours.Text + "','" +

gEducationPlanDeleteYear.Text + "','" +

specId + "','" +

subjectId + "')";

Console.WriteLine(sql);

conn.Open();

MySqlCommand cmd = new MySqlCommand(sql, conn);

try

{

cmd.ExecuteNonQuery();

}

catch (MySqlException ex)

{

error = true;

MessageBox.Show("Ошибка удаления\n" + ex.ToString());

}

}

finally

{

conn.Close();

}

if (!error) MessageBox.Show("Запись успешно удалена!");

}

}

public bool IsSplittable(int groupId, int subjectId)

{

int students = 0, themes = 0;

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call groupCount('" + groupId +"')";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

try

{

students = Convert.ToInt32(rdr[0].ToString());

}

catch (Exception ex)

{

// Console.WriteLine(ex.ToString());

}

}

rdr.Close();

}

finally

{

conn.Close();

}

conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call themeCount('" + subjectId + "')";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

try

{

themes = Convert.ToInt32(rdr[0].ToString());

}

catch (Exception ex)

{

// Console.WriteLine(ex.ToString());

}

}

rdr.Close();

}

finally

{

conn.Close();

}

if (themes >= students) return true;

else return false;

}

private class S\_S

{

public string student;

public string theme;

}

private class BigData

{

public int student\_id { get; set; }

public string student { get; set; }

public int theme\_id { get; set; }

public string theme { get; set; }

}

public void DistributeThemes()

{

if (!done)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT id, name FROM groups ORDER BY id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list.Add(it);

}

catch (Exception ex)

{

// Console.WriteLine(ex.ToString());

}

gAutoGroups.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT id, name FROM subjects ORDER BY id";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list2.Add(it);

}

catch (Exception ex)

{

// Console.WriteLine(ex.ToString());

}

gAutoSubject.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

}

else

{

List<S\_S> s\_s = new List<S\_S>();

int groupId = 0, subjectId = 0;

foreach (Data d in list)

if (d.name == gAutoGroups.Text)

groupId = d.id;

foreach (Data d in list2)

if (d.name == gAutoSubject.Text)

subjectId = d.id;

bool error = false;

if (IsSplittable(groupId,subjectId))

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "call show\_engaged\_themes()";

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

string students = null;

int i = 0;

while (rdr.Read())

{

try

{

string studentName = rdr[1].ToString();

if (gAutoGroups.Text == rdr[0].ToString() &&

gAutoSubject.Text == rdr[2].ToString())

{

S\_S it = new S\_S();

it.student = studentName;

//error = true;

students += ++i + " " + studentName + "\n";

it.theme = rdr[3].ToString();

s\_s.Add(it);

//break;

}

}

catch (Exception ex)

{

// Console.WriteLine(ex.ToString());

}

}

if ( i != 0 )

{

students += "\nУ этих студентов уже зарезервированы темы!";

MessageBox.Show(students,i + " студентов уже имеют темы");

}

rdr.Close();

if (!error)

{

List<BigData> BDlist = new List<BigData>();

sql = "call autoSplit(@group\_id,@subject\_id)";

conn = new MySqlConnection(connStr);

cmd = new MySqlCommand(sql,conn);

cmd.Parameters.AddWithValue("@group\_id", groupId);

cmd.Parameters.AddWithValue("@subject\_id", subjectId);

conn.Open();

rdr = cmd.ExecuteReader();

while (rdr.Read())

{

string studentTheme = rdr[3].ToString();

string studentName = rdr[1].ToString();

BigData it = new BigData();

it.student\_id = Convert.ToInt32(rdr[0].ToString());

it.student = studentName;

it.theme\_id = Convert.ToInt32(rdr[2].ToString());

it.theme = studentTheme;

BDlist.Add(it);

}

for (int n = 0; n < BDlist.Count - 1; ++n)

{

BDlist.RemoveAll(

it => s\_s.Exists(

it2 => ((it2.student == it.student) || (it2.theme == it.theme))

)

);

sql = "call add\_engaged\_themes(@myStudents\_id," +

"@myStudents\_Groups\_id,@myThemes\_id," +

"@myThemes\_Subjects\_id,@myMark)";

conn = new MySqlConnection(connStr);

conn.Open();

cmd = new MySqlCommand(sql,conn);

//Console.WriteLine("Count - " + BDlist.Count);

int std\_id = 0, thm\_id = 0;

string std = "", thm = "";

try

{

std\_id = BDlist[0].student\_id;

std = BDlist[0].student;

thm\_id = BDlist[0].theme\_id;

thm = BDlist[0].theme;

} catch(Exception ex)

{

Console.WriteLine(ex.ToString());

}

cmd.Parameters.AddWithValue("@myStudents\_id", std\_id);

cmd.Parameters.AddWithValue("@myStudents\_Groups\_id", groupId);

cmd.Parameters.AddWithValue("@myThemes\_id", thm\_id);

cmd.Parameters.AddWithValue("@myThemes\_Subjects\_id", subjectId);

cmd.Parameters.AddWithValue("@myMark",0);

try

{

cmd.ExecuteNonQuery();

} catch(Exception ex)

{

Console.WriteLine(ex.ToString());

}

S\_S s = new S\_S();

s.student = std;

s.theme = thm;

s\_s.Add(s);

}

conn.Close();

MessageBox.Show("Свободные темы были распределены");

}

}

finally

{

conn.Close();

}

}

else

{

error = true;

MessageBox.Show("Невозможно распределить темы.\nСтудентов больше чем существует тем!");

}

//if (!error) MessageBox.Show("Темы успешно распределены между студентами!");

}

}

private void ActionButton\_Click(object sender, RoutedEventArgs e) // все события

{

done = true;

if (type == "Распределить") DistributeThemes();

if (type == "Темы" && action == "Добавить") ThemesAdd(); // +

if (type == "Темы" && action == "Удалить") ThemesDelete(); // +

if (type == "Специальности" && action == "Добавить") SpecsAdd(); // +

if (type == "Специальности" && action == "Удалить") SpecsDelete(); // +

if (type == "Студенты" && action == "Добавить") StudentsAdd(); // +

if (type == "Студенты" && action == "Удалить") StudentsDelete(); // +

if (type == "Группы" && action == "Добавить") GroupsAdd(); // +

if (type == "Группы" && action == "Удалить") GroupsDelete(); // +

if (type == "Предметы" && action == "Добавить") SubjectsAdd(); // +

if (type == "Предметы" && action == "Удалить") SubjectsDelete(); // +

if (type == "Учебный план" && action == "Добавить") EduPlanAdd(); // +

if (type == "Учебный план" && action == "Удалить") EduPlanDelete(); // +

// Занятые темы // +

// Занятые темы // +

this.Hide();

this.Close();

}

private void gThemesDeleteName\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

MySqlConnection conn = new MySqlConnection(connStr);

try

{

int myId = 0;

foreach (Data d in list)

{

Console.WriteLine(d.id + " - " + d.name + " ? " + gThemesDeleteSubject.Text);

if (gThemesDeleteSubject.Text == d.name)

{

Console.WriteLine("--> " + d.id + " - " + d.name);

myId = d.id;

}

}

conn.Open();

string sql = "SELECT Id, name FROM themes " +

"WHERE subjects\_id = " +

myId + " ORDER BY Id";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

gThemesDeleteName.Items.Clear();

while (rdr.Read())

{

try

{

Data it = new Data();

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list2.Add(it);

gThemesDeleteName.Items.Add(it.name);

}

catch (Exception ex)

{

MessageBox.Show(

"Произошла ошибка\n"

+ ex.ToString());

}

}

rdr.Close();

}

catch (MySqlException ex)

{

MessageBox.Show("Произошла ошибка\n" + ex.ToString());

}

}

private void gStudentDeleteStudent\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

int groups\_id = 0;

foreach (Data d in list)

{

Console.WriteLine(d.id + d.name);

if (d.name == gStudentDeleteGroup.Text)

groups\_id = d.id;

}

MySqlConnection conn = new MySqlConnection(connStr);

try

{

conn.Open();

string sql = "SELECT id, surname from students " +

"where groups\_id = '" + groups\_id +

"' ORDER BY Id";

Console.WriteLine(sql);

MySqlCommand cmd = new MySqlCommand(sql, conn);

MySqlDataReader rdr = cmd.ExecuteReader();

gStudentDeleteStudent.Items.Clear();

while (rdr.Read())

{

Data it = new Data();

try

{

it.id = Convert.ToInt32(rdr[0].ToString());

it.name = rdr[1].ToString();

list2.Add(it);

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

}

gStudentDeleteStudent.Items.Add(it.name);

}

rdr.Close();

}

finally

{

conn.Close();

}

}

}

}

**Файл addForm.xaml.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace MyApp

{

/// <summary>

/// Логика взаимодействия для addForm.xaml

/// </summary>

public partial class addForm : Window

{

public addForm()

{

InitializeComponent();

}

public bool checkString(string str)

{

foreach (char c in str)

if (!System.Text.RegularExpressions.Regex.IsMatch(c.ToString(), "^[a-zA-Z]"))

return false;

return true;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

bool ok = true;

//оптимизировать - достаточно регуляркой всю строку проверять, а один символ проверять неэффективно

if (ok) ok = checkString(LoginEdit.Text.ToString());

if (ok) ok = checkString(PasswordEdit.Text.ToString());

if (ok) ok = checkString(NameEdit.Text.ToString());

if (ok) ok = checkString(SurnameEdit.Text.ToString());

bool error = false;

try

{

common.nRole = RoleBox.SelectedValue.ToString();

common.nGender = GenderBox.SelectedValue.ToString();

}

catch (Exception ex)

{

Console.WriteLine(ex.ToString());

error = true;

}

common.nGender = GenderBox.Text.ToString();

common.nUsername = LoginEdit.Text.ToString();

common.nPassword = PasswordEdit.Text.ToString();

common.nName = NameEdit.Text.ToString();

common.nSurname = SurnameEdit.Text.ToString();

if (common.nUsername.ToString() != "" &&

common.nPassword.ToString() != "" &&

common.nName.ToString() != "" &&

common.nSurname.ToString() != "" &&

!error && ok)

{

common.wannaToCreateUser = true;

Hide();

Close();

}

else MessageBox.Show("Некоторые поля не заполнены или заполнены не верно");

}

}

}