СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная курсовая работа выполнена в соответствии с заданием на курсовое проектирование. Темой работы является «Система управления «Центр детского творчества».

В современном мире стремительными темпами развиваются информационные технологии и те сферы человеческой деятельности, которые с ними связаны. С каждым годом всё больший и больший объём информации обрабатывается при помощи современных и продолжающих совершенствоваться персональных компьютерах.

Любая организация нуждается в своевременном доступе к информации. Ценность информации в современном мире очень высока. Роль распорядителей информации в современном мире чаще всего выполняют базы данных. Базы данных обеспечивают надежное хранение информации, структурированном виде и своевременный доступ к ней. Практически любая современная организация нуждается в базе данных, удовлетворяющей те или иные потребности по хранению, управлению и администрированию данных.

Главная цель, которая стоит перед написанием курсового проекта – это приобретение практических навыков проектирования базы данных «Центр детского творчества» с использованием программ СУБД. В ходе выполнения данного курсового проекта необходимо будет разработать информационную базу данных и программное обеспечение для работы совместно с базой данных для центра детского творчества, которые помогут пользователю легко найти нужную информацию о сотруднике, детях, группах и событиях в любом момент времени.

1 ПОСТАНОВКА ЗАДАЧИ

1.1. Описание предметной области

База данных — важнейший компонент любой информационной системы. База данных позволяет структурировано хранить большие объемы информации конкретного предприятия, что значительно рационализирует ведение отчетов и создание архивов. Оптимизированные базы данных значительно увеличивают производительность, построенных на их использовании, программ.

С развитием информационных технологий и предпринимательства, актуальность использования баз данных значительно увеличилось. Успешные и крупные компании не могут представить свой бизнес без четко построенной информационной системы. Базы данных, построенные на SQL Server, отвечают высоким требованиям производительности и безопасности.

Предметом курсового проекта является проектирование информационной базы данных и создание программного обеспечения для работы совместно с базой данных для центра детского творчества.

На сегодняшний день существует большое количество аналогов. Для сравнения функционального и визуального обеспечения программы «Центр детского творчества», были выбраны программы «Хеликс: Детский Центр» и «CRM-система для автоматизации детских центров Отмечалка».

«Хеликс: Детский Центр» - компьютерная программа, разработанная специально для детских развивающих центров, которая поможет автоматизировать все рутинные процессы по учету и управлению бизнесом.

"Хеликс: Детский Центр" разработан на платформе "1С: Предприятие 8.3". Из плюсов программы можно выделить то что она максимально простая и удобная в работе, а также в ней реализованы множество функциональных возможностей. Из минусов можно отметить, что «Хеликс: Детский Центр» является платной программой, для её установки требуется выезд специалиста, и гарантия на обслуживание составляет всего 3 месяца.

Рисунок 1.1 – Хеликс: Детский Центр»

«CRM-система для автоматизации детских центров Отмечалка» - это система автоматизации, предоставляющая полный набор инструментов, которые помогут раскрыть потенциал детского центра, повысить его прибыльность и конкурентоспособность.

Из плюсов это программы можно отметить то что в ней есть бесплатный 14 дневный период, постоянное обновление программы и высокая надёжность хранения данных.

В минусы этой программы также входят можно записать необходимость платить за её использование, а также необходимость подключения к сети интернет.

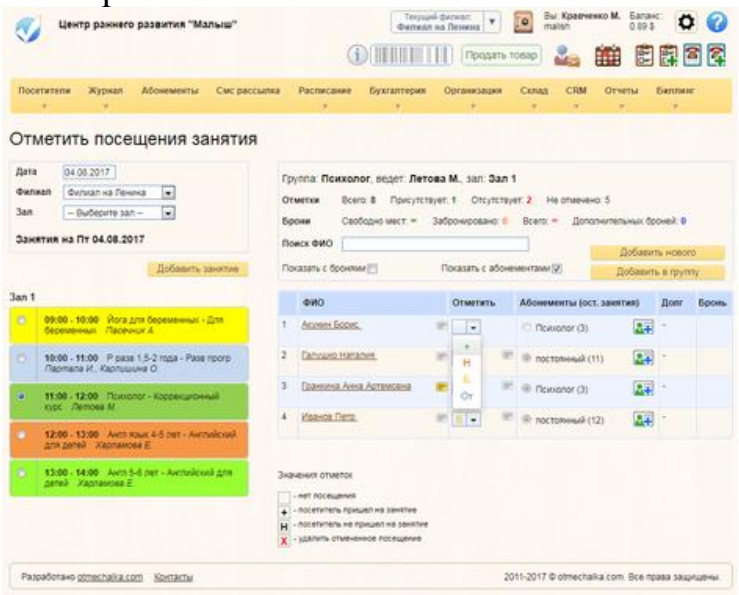


Рисунок 1.2 – «CRM-система Отмечалка»

Сравнительный анализ показал, что «Центр детского творчества», разрабатываемый в рамках курсового проектирования, по основным характеристикам не уступает существующим аналогам.

1.2. Цели и задачи проектирования

Целью данного курсового проекта является реализация программной системы «Цент детского творчества» для автоматизации деятельности и обеспечения хранения, накопления и предоставления информации

Задачи проекта:

- Анализ предметной области;
- Построение инфологической (концептуальной) модели предметной области;
- Разработка логической структуры базы данных;
- Разработка физической структуры базы данных;
- Организация базы данных;

- Разграничение прав доступа.
- Разработка пользовательского интерфейса;
- Реализация проекта в конкретной среде программирования;
- Наполнение базы данных;

Входной информацией в создаваемой программе будут являться запросы пользователей.

Выходной информацией является подробная информация о детях, сотрудниках, группах и событиях представленная в виде таблиц.

2 ПРОЕКТНАЯ ЧАСТЬ

2.1. Проектирование схемы данных

Проектирование базы данных осуществляется в три этапа: концептуальное (инфологическое) проектирование, логическое проектирование и физическое проектирование;

Цель инфологического этапа проектирования состоит в получении семантических (концептуальных) моделей, отражающих предметную область и информационные потребности пользователей. В качестве инструмента для построения семантических моделей данных на этапе инфологического проектирования является неформальная модель "Сущность-Связь" (ER-модель - Entity-Relationship). Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов.

Основными понятиями ER-модели являются сущность, связь и атрибут.

Сущность (объект) - это реальный или представляемый объект предметной области, информация о котором должна сохраняться и быть доступна. Различают такие понятия, как тип сущности и экземпляр сущности. Понятие тип сущности относится к набору однородных предметов, событий, личностей, выступающих как единое целое. Экземпляр сущности относится к конкретной вещи в наборе. В диаграммах ER-модели сущность представляется в виде прямоугольника (в нотации Баркера), содержащего имя сущности.

Выделим базовые сущности данного курсового проекта:

- **Дети**. Атрибуты: информация о детях, родители.
- **События**. Атрибуты – описание, дата проведения, заметки, цена.
- **Группы**. Атрибуты – сотрудник, класс, расписание, размер группы, цена.
- **Сотрудники**. Атрибуты – ФИО, дата рождения, квалификация.

Родители и **Информацию о детях** будем рассматривать как связь с детьми. Атрибуты родителей: ФИО мамы, ФИО папы, дата рождения мамы, дата рождения папы. Атрибуты информации о детях: ФИО, дата рождения, адрес.

Класс будем рассматривать как связь с группой. Атрибуты класса – название, описание, заметки, категория. А **Категории** рассматриваем как связь с классом. Атрибуты категории – название, описание.

Лист в данной работе рассматривается как связь между такими сущностями как дети, события и группы.

Атрибут - поименованная характеристика сущности, определяющая его свойства и принимающая значения из некоторого множества значений. Каждый атрибут обеспечивается именем, уникальным в пределах сущности.

Связь (Relationship) - это поименованная графически изображаемая ассоциация, устанавливаемая между сущностями и представляющая собой абстракцию набора отношений, которые систематически возникают между различными видами предметов в реальном мире. Большинство связей относятся к категории бинарных и имеют место между двумя сущностями.

Среди бинарных связей существуют три фундаментальных вида связи: один-к-одному (1:1), один-ко-многим (1:M), многие-ко-многим (M:M). Связь один-к-одному (1:1) существует, когда один экземпляр одной сущности связан с единственным экземпляром другой сущности. Связь один-ко-многим (1:M) имеет место, когда один экземпляр одной сущности связан с одним или более экземпляром другой сущности и каждый экземпляр второй сущности связан только с одним экземпляром первой сущности. Связь многие-ко-многим (M:N) существует, когда один экземпляр одной сущности связан с одним или более экземпляром другой сущности и каждый экземпляр второй сущности связан с одним или более экземпляром первой сущности.

В данной курсовой работе предоставлены все три вида связей. Инфологическая (концептуальная) модель предметной области представлена в Приложении В.

Функциональные возможности:

- ведение БД (запись, чтение, модификация, удаление);
- обеспечение логической непротиворечивости БД;
- реализация наиболее часто встречающихся запросов в готовом виде;
- предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

Запросы:

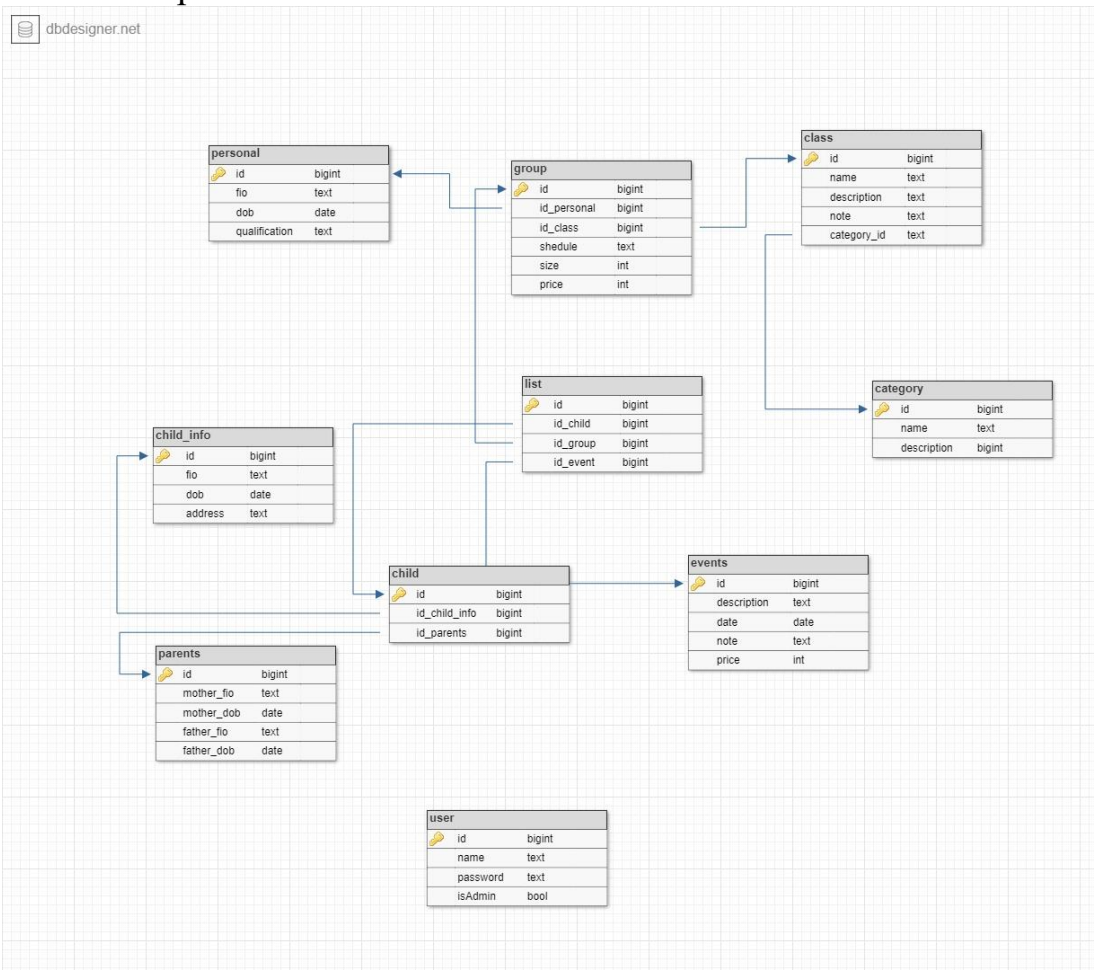
- получение списка всех групп, включая информацию о детях и классах;
- получение списка всех событий, включая информацию о детях;
- получение списка персонала;
- получение списка детей;
- получение списка полной информации о детях;
- получение списка родителей;
- получение списка классов;
- получение списка категорий;
- удаление данных из выше перечисленных списков;
- редактирование данных из выше перечисленных списков;
- добавление данных в выше перечисленные списки;

Логическое проектирование это проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, задание их имен и типов, а также некоторых количественных характеристик (например, длины поля).

При проектировании логической структуры БД, осуществляется преобразование исходной инфологической модели в модель данных,

поддерживаемую конкретной СУБД, и проверка адекватности, полученной логической модели отображаемой предметной области.

Схема базы «Системы управления «Центр детского творчества» представлена в приложении Б.



Описание и структура спроектированной базы данных:

Категории (id (счетчик (Длинное целое)), название (Текстовый), описание(Текстовый));

Дети (id (счетчик (Длинное целое)), id информации о ребёнке (счетчик (Длинное целое)), id родителей (счетчик (Длинное целое)));

Информация о детях (id (счетчик (Длинное целое)), ФИО (Текстовый), Дата рождения (Дата), адрес (Текстовый));

Класс (id (счетчик (Длинное целое)), название (Текстовый), описание (Текстовый), заметки (Текстовый), категория (счетчик (Длинное целое)));

События (id (счетчик (Длинное целое)), описание (Текстовый), дата проведения(Дата/Время), заметки (Текстовый), цена (Числовой (Длинное целое)));

Группы (id (счетчик (Длинное целое)), учитель (счетчик (Длинное целое)), тип группы (счетчик (Длинное целое)), расписание (Текстовый), размер группы (Числовой (Длинное целое)), цена (Числовой (Длинное целое)));

Список (id (счетчик (Длинное целое)), id ребёнка (счетчик (Длинное целое)), id группы (счетчик (Длинное целое)) id события (счетчик (Длинное целое)));

Родители (id (счетчик (Длинное целое)), ФИО матери (Текстовый), Дата рождения матери (Дата), ФИО отца (Текстовый), Дата рождения отца (Дата));

Персонал (id (счетчик (Длинное целое)), ФИО (Текстовый), Дата рождения (Дата), квалификация (Текстовый));

Удаленные события (id (счетчик (Длинное целое)), описание (Текстовый), дата проведения(Дата/Время), заметки (Текстовый), цена (Числовой (Длинное целое)));

Удаленные Группы (id (счетчик (Длинное целое)), учитель (счетчик (Длинное целое)), тип группы (счетчик (Длинное целое)), расписание (Текстовый), размер группы (Числовой (Длинное целое)), цена (Числовой (Длинное целое)));

Пользователь (id (счетчик (Длинное целое)), логин (Текстовый), пароль (Текстовый), если Админ (Числовое (Короткое целое)), последняя дата входа (Дата/Время), последняя дата выхода (Дата/Время));

2.2. Обоснование выбора технологий

C# — объектно-ориентированный язык программирования. C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Основные достоинства C#:

- C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности — как FCL, так и CLR;
- C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
- C# является мощным объектным языком с возможностями наследования и универсализации;
- C# является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от C++ к C#;
- сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются

такие опасные свойства C++ как указатели, адресация, разыменование, адресная арифметика;

- благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;
- мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;
- реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#

2.3. Описание среды реализации

СУБД MySql — предоставляет мощные средства для доступа, настройки, администрирования, разработки всех компонентов базы данных и управления ими. MySql — это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySql — свободно распространяемая система. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД. Работать с MySql можно не только в текстовом режиме, но и в графическом. Существует очень популярный визуальный интерфейс для работы с этой СУБД — PhpMyAdmin. Этот интерфейс позволяет значительно упростить работу с базами данных в MySql.

PhpMyAdmin позволяет пользоваться всеми достоинствами браузера, включая прокрутку изображения, если оно не умещается на экран. Многие из базовых SQL-функций работы с данными в PhpMyAdmin сведены к интуитивно понятным интерфейсам и действиям, напоминающим переход по ссылкам в Internet.

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода,

так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. VisualStudio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (например, Subversion и VisualSourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода). Главным преимуществом Visual Studio 2017 является производительность. Обеспечивает возможность создания разнообразных приложений на основе одного набора навыков.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1. Физическая структура базы данных

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

Для проектирования базы данных для «Системы управления «Центр детского творчества» была выбрана СУБД MySQL. Для хранения данных в этой СУБД используются таблицы. В них хранится вся информация о предметной области. Наша база данных включает несколько взаимосвязанных таблиц. Объекты, которые были описаны при построении инфологической модели предметной области, в базе данных являются таблицами.

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	name	TEXT		☐	☐	☐	No default
3	description	TEXT		☐	☐	☐	No default

Рисунок 3.1 – Таблица «Категории»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	id_child_info	BIGINT	20	☐	☐	☐	No default
3	id_parents	BIGINT	20	☐	☐	☐	No default

Рисунок 3.2 – Таблица «Дети»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	fio	TEXT		☐	☐	☐	No default
3	dob	DATE		☐	☐	☐	No default
4	address	TEXT		☐	☐	☐	No default

Рисунок 3.3 – Таблица «Информация о детях»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	name	TEXT		☐	☐	☐	No default
3	description	TEXT		☐	☐	☐	No default
4	note	TEXT		☐	☐	☐	No default
5	category_id	TEXT		☐	☐	☐	No default

Рисунок 3.4 – Таблица «Классы»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	description	TEXT		☐	☐	☐	No default
3	date	DATE		☐	☐	☐	No default
4	note	TEXT		☐	☐	☐	No default
5	price	INT	11	☐	☐	☐	No default

Рисунок 3.5 – Таблица «События»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	id_personal	BIGINT	20	☐	☐	☐	No default
3	id_class	BIGINT	20	☐	☐	☐	No default
4	shedule	TEXT		☐	☐	☐	No default
5	size	INT	11	☐	☐	☐	No default
6	price	INT	11	☐	☐	☐	No default

Рисунок 3.6 – Таблица «Группы»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	id_child	BIGINT	20	☐	☐	☐	No default
3	id_group	BIGINT	20	☐	☐	☐	No default
4	id_event	BIGINT	20	☐	☐	☐	No default

Рисунок 3.7 – Таблица «Лист»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	mother_fio	TEXT		☐	☐	☐	No default
3	mother_dob	DATE		☐	☐	☐	No default
4	father_fio	TEXT		☐	☐	☐	No default
5	father_dob	DATE		☐	☐	☐	No default

Рисунок 3.8 – Таблица «Родители»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	fio	TEXT		☐	☐	☐	No default
3	dob	DATE		☐	☐	☐	No default
4	qualification	TEXT		☐	☐	☐	No default

Рисунок 3.9 – Таблица «Персонал»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	INT	11	☐	☐	☐	AUTO_INCREMENT
2	description	TEXT		☐	☐	☐	No default
3	date	DATE		☐	☐	☐	No default
4	note	TEXT		☐	☐	☐	No default
5	price	INT	11	☐	☐	☐	No default

Рисунок 3.10 – Таблица «Удаленные события»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	INT	11	☐	☐	☐	No default
2	id_personal	BIGINT	20	☐	☐	☐	No default
3	id_class	BIGINT	20	☐	☐	☐	No default
4	shedule	TEXT		☐	☐	☐	No default
5	size	INT	11	☐	☐	☐	No default
6	price	INT	11	☐	☐	☐	No default

Рисунок 3.11 – Таблица «Удаленные группы»

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	id	BIGINT	20	☐	☐	☐	AUTO_INCREMENT
2	name	TEXT		☐	☐	☐	No default
3	password	TEXT		☐	☐	☐	No default
4	isAdmin	TINYINT	1	☐	☐	☐	No default
5	lastSign	DATETIME		☐	☑	☐	NULL
6	lastLogout	DATETIME		☐	☑	☐	NULL

Рисунок 3.12 – Таблица «Пользователи»

Таблицы соответствуют трём нормальным формам. Для связи между разными таблицами используется один и тот же ключ в разных таблицах и специально связующие таблицы. Все таблицы идентифицируются числом.

Существует несколько способов реализации управления базой данных, в частности, любое из указанных действий можно выполнить тремя способами:

- через раздел СУБД «Таблицы», производя действия по изменению, добавлению или удалению непосредственно в таблице;
- через раздел СУБД «Формы», выполняя необходимые действия в таблице через интерфейс формы;
- через раздел СУБД «Запросы», выполняя запросы на обновление, добавление или удаление данных.

Существует 3 способа ввода данных: ввод с клавиатуры; сохранение

данных, сформированных иными программными средствами; импорт из других источников. В нашей базе данных мы использовали ввод с клавиатуры.

При проектировании нашей базы данных ввод информации осуществлялся при помощи таблиц. Корректировка данных возможна в этих же таблицах.

Учитывая, что наш проект предполагает осуществление поступления данных в базу через windows-приложение - для удобства работы с нашей базой данных мы реализовали специальные формы с таблицами, частично эмулирующие работу бд.

При работе с базой данных, у пользователя могут возникать следующие потребности:

- Нахождение нужной информации
- Отображение и просмотр этой информации

В данном курсовом проекте информационные потребности пользователей реализуются с помощью запросов.

Запрос - это требование на получение определенной информации. Запросы позволяют сфокусировать внимание именно для тех данных, которые нужны для решения текущей задачи.

Используя запросы, можно проверять данные любым образом, который пользователь в состоянии представить. Можно отобрать таблицы, поля и записи, содержащие необходимые для просмотра, подведения итогов или использования в вычислениях данные; отсортировать их; создать отчеты и формы для отображения указанной нами информации и даже создать диаграммы для наглядного представления данных.

Результат работы запроса называется выборкой. Выборка не сохраняется в базе данных; она создается заново каждый раз при выполнении запроса и уничтожается при его закрытии.

Существуют следующие виды запросов:

1. Запрос для отбора данных по заданным сложным условиям из одной или нескольких таблиц баз данных, с группировкой данных для расчета итогов, с показом результатов выполнения запроса в виде таблицы, либо с использованием его для форм и отчетов. После редактирования данных в таблице запроса данные таблиц базы могут обновляться (с некоторыми ограничениями).
2. Перекрестный запрос с формированием двухмерной итоговой таблицы, с группировкой по двум выражениям, одно из которых становится заголовком строки, другое - заголовком столбца.
3. Запрос на создание новой таблицы.
4. Запросы на изменение данных:
    - обновление данных - команда занесения общих изменений в группу записей одной или нескольких таблиц;

- добавление данных - команда добавления группы записей из одной или нескольких таблиц в конец одной или нескольких таблиц;
- удаление данных - команда удаления группы записей из одной или нескольких таблиц.

Триггер — это хранимая процедура, которая не вызывается непосредственно, а исполняется при наступлении определенного события (вставка, удаление, обновление строки).

В данном курсовом проекте два триггера. Оба триггера срабатывают при событии удаления в таблицах «Группы» и «События» и добавляют копию удалённой записи в отдельные таблицы «Удалённые группы» и «Удалённые события».

Запрос на создание триггера для таблицы «Группы»:

```
CREATE DEFINER=`root`@`%` TRIGGER `group_after_delete` AFTER
DELETE ON `group` FOR EACH ROW BEGIN
INSERT INTO `removed_group` Set id = OLD.id, id_personal =
OLD.id_personal, id_class = OLD.id_class, shedule = OLD.shedule,
size = OLD.size, price = OLD.price;
END
```

Запрос на создание триггера для таблицы «События» :

```
CREATE DEFINER=`root`@`%` TRIGGER `events_after_delete` AFTER
DELETE ON `events` FOR EACH ROW BEGIN
INSERT INTO `removed_events` Set id = OLD.id, description =
OLD.description, date = OLD.date, note = OLD.note, price =
OLD.price;
END
```

Хранимые процедуры – это способ инкапсуляции повторяющихся действий. В хранимых процедурах можно объявлять переменные, управлять потоками данных, а также применять другие техники программирования.

В данном курсовом проекте используется 2 хранимые процедуры.

Создание хранимой процедуры для обновления даты выхода из системы:

```
CREATE DEFINER=`root`@`%` PROCEDURE `updateLastLogout`(
IN `var1` INT
,
IN `var1` INT

)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
UPDATE `user` SET `lastLogout`=CURDATE() WHERE `id`=var1
```

Создание хранимой процедуры для обновления даты авторизации пользователя:

```
CREATE DEFINER=`root`@`%` PROCEDURE `updateUserSign`(
IN `var1` INT
,
IN `var1` INT

)
LANGUAGE SQL
```

```
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
UPDATE `user` SET `lastSign`=CURDATE() WHERE `id`=var1
```

Первая процедура при её вызове обновляет дату и время авторизации пользователя. Вторая процедура обновляет дату и время выхода из системы пользователя.

3.2. Структура приложения

При проектировании и разработке любой базы данных интерфейс играет важную роль. Он представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными компонентами базы данных.

Работа с базой начинается с авторизации:

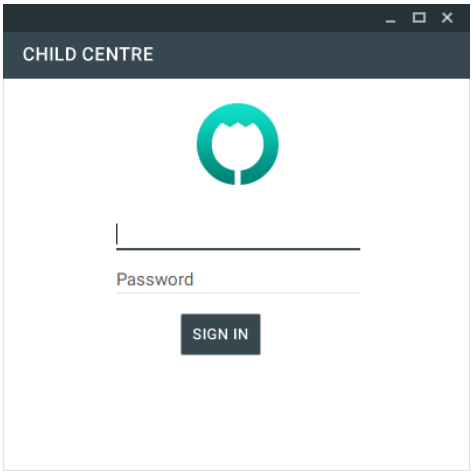


Рисунок 3.13 – Меню авторизации

При неверно введенных данных появляется окно с ошибкой

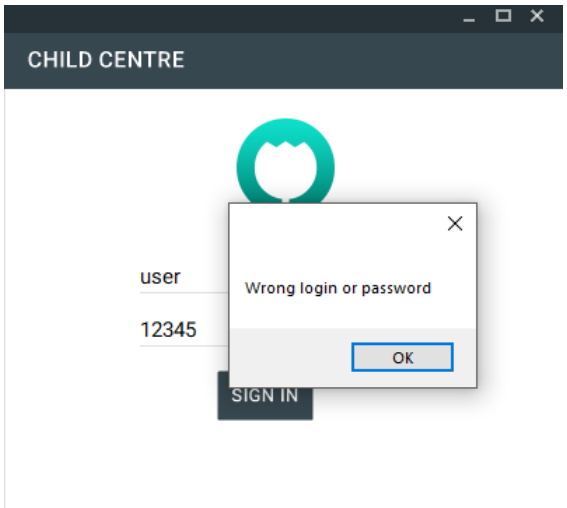


Рисунок 3.14 – Ошибка при авторизации

После авторизации в зависимости от роли пользователя можно попасть в форму администратора и форму пользователя, которые предназначены для предоставления пользователю удобной навигации по всем разделами базы данных.

Окно с формами для работы менеджером включает в себя разделы «Группы», «Дети», «События», «Удаленные события», «Удаленные группы» и кнопку выхода из формы. Так же на форме присутствует форма поиска и сортировка данных по таблице.

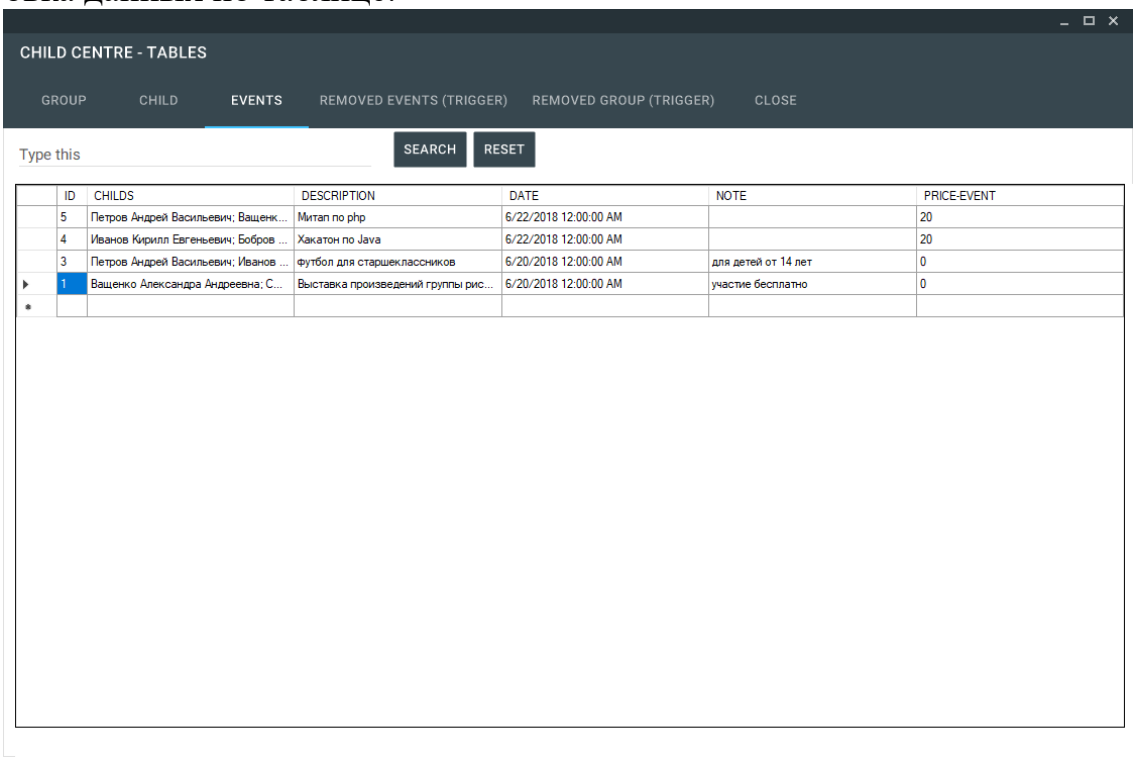
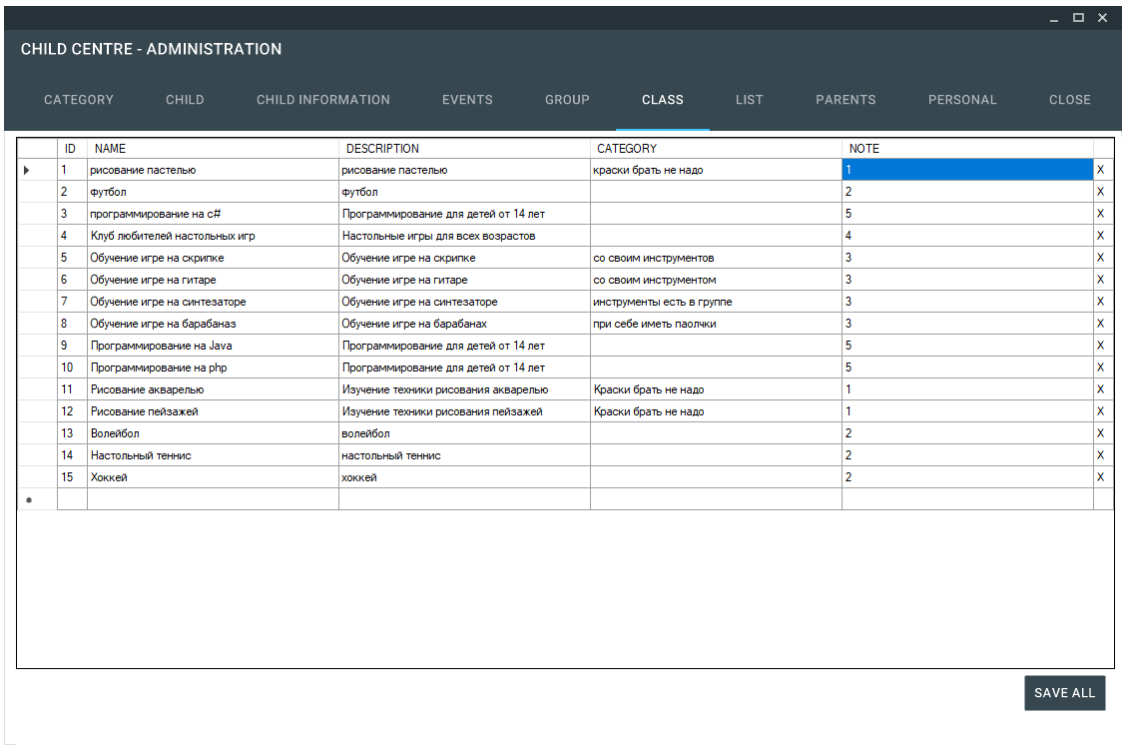


Рисунок 3.15 – Окно менеджера

Окно с формами для работы администратора включает в себя разделы «Категории», «Группы», «Дети», «Информация о детях», «События», «Классы», «Список», «Родители», «Персонал» и кнопку выхода из формы. Так же на форме присутствует сортировка данных по таблице.



ID	NAME	DESCRIPTION	CATEGORY	NOTE	
1	рисование пастелью	рисование пастелью	краски брать не надо	1	X
2	футбол	футбол		2	X
3	программирование на c#	Программирование для детей от 14 лет		5	X
4	Клуб любителей настольных игр	Настольные игры для всех возрастов		4	X
5	Обучение игре на скрипке	Обучение игре на скрипке	со своим инструментов	3	X
6	Обучение игре на гитаре	Обучение игре на гитаре	со своим инструментом	3	X
7	Обучение игре на синтезаторе	Обучение игре на синтезаторе	инструменты есть в группе	3	X
8	Обучение игре на барабаназ	Обучение игре на барабанах	при себе иметь паолчки	3	X
9	Программирование на Java	Программирование для детей от 14 лет		5	X
10	Программирование на php	Программирование для детей от 14 лет		5	X
11	Рисование акварелью	Изучение техники рисования акварелью	Краски брать не надо	1	X
12	Рисование пейзажей	Изучение техники рисования пейзажей	Краски брать не надо	1	X
13	Волейбол	волейбол		2	X
14	Настольный теннис	настольный теннис		2	X
15	Хоккей	хоккей		2	X

Рисунок 3.16 – Окно администратора

Добавление и редактирование пользователей происходит через внесение или редактирование данных в таблице и нажатия кнопки «Sava all». Удаление записей происходит по нажатию на «x» в таблице.

4 ТЕСТИРОВАНИЕ

Тестирование (testing) программного обеспечения (ПО) - это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. В более широком смысле, тестирование ПО - это техника контроля качества программного продукта, включающая в себя проектирование тестов, выполнение тестирования и анализ полученных результатов.

Очень часто современные программные продукты разрабатываются в сжатые сроки и при ограниченных бюджетах проектов. Программирование сегодня перешло из разряда искусства в разряд ремесел для многих миллионов специалистов. Но, к сожалению, в такой спешке разработчики зачастую игнорируют необходимость обеспечения защищённости своих продуктов, подвергая тем самым пользователей неоправданному риску. Контроль качества (тестирование) считается важным в процессе разработки ПО, потому что обеспечивает безопасность, надёжность, удобство создаваемого продукта. В настоящее время существует великое множество подходов и методик к решению задачи тестирования ПО, но эффективное тестирование сложных программных систем - процесс творческий, не сводящийся к следованию строгим и чётким правилам.

В рамках курсового проекта, программное средство будет протестировано одним из распространенных видов тестирования – функциональным тестированием.

Функциональное тестирование (functional testing) – вид тестирования, направленный на проверку корректности работы функциональности приложения (корректность реализации функциональных требований.

Результаты тестирования программного продукта представлены в таблицах 4.1 – 4.3.

Таблица 4.1 – Результаты тестирования формы пользователя

Номер	Условия	Действие	Ожидаемый результат	Итог
1	2	3	4	5
1	При нажатии на вкладку	Нажатие на вкладку	Загрузка таблицы с данными	пройден
2	При поиске	Ввод данных в строку поиска и нажатие на кнопку «Search»	Обновление таблицы в соответствии с условиями поиска	пройден

Продолжение таблицы 4.1

3	При сбросе данных поиска	Нажатие на кнопку «Reset»	Обновление таблицы	пройден
4	При сортировке данных	Нажатие на заголовки таблицы	Сортировка данных в таблице в соответствии с выбранными параметрами	пройден

Таблица 4.2 – Результаты тестирования формы авторизации

Номер	Условия	Действие	Ожидаемый результат	Итог
1	2	3	4	5
1	При верном вводе данных для авторизации администратора и нажатии на кнопку «Sign in»	Ввод данных для авторизации и нажатие на кнопку «Sign in»	Авторизация и открытие нового окна с формами для работы от лица администратора	пройден
2	При верном вводе данных для авторизации пользователя и нажатии на кнопку «Sign in»	Ввод данных для авторизации и нажатие на кнопку «Sign in»	Авторизация и открытие нового окна с формами для работы от лица пользователя	пройден
3	При неверном вводе данных для авторизации и нажатии на кнопку «Sign in»	Ввод данных для авторизации и нажатие на кнопку «Sign in»	Открытие нового окна с выводом ошибки	пройден

Таблица 4.3 – Результаты тестирования формы администратора

Номер	Условия	Действие	Ожидаемый результат	Итог
1	2	3	4	5
1	При нажатии на вкладку	Нажатие на вкладку	Загрузка таблицы с данными, отображение кнопки сохранения данных в таблицу	пройден
2	При добавлении данных	Ввод данных в таблицу и нажатие кнопки сохранить	Добавление данных в бд и обновление таблицы	пройден
3	При редактировании данных	Редактирование данных в таблице и нажатие кнопки сохранить	Обновление данных в бд и обновление таблицы	пройден
4	При удаление данных	Удаление данных в таблице	Удаление данных в бд и обновление таблицы	пройден

Выше были приведены основные проверки форм программного средства. По итоговому результату можно сделать вывод о том, что программное средство выполняет все необходимые функции.

5 ОПИСАНИЕ ПРИМЕНЕНИЯ

Программный продукт «Система управления «Центр детского творчества» поможет пользователю легко найти нужную информацию о сотруднике, детях, группах и событиях в любом момент времени, автоматизировать все рутинные процессы по учету и управлению данными.

Сфера применения программы «Система управления «Центр детского творчества» это администрирование детских развивающих центров, детских творческих клубов, центров раннего развития и предприятий подобной направленности.

С помощью программы «Центр детского творчества» можно будет выполнять следующие функции:

- Авторизация пользователей;
- Вести клиентскую базу детей с учетом их сопровождающих (родители, родственники);
- Вести клиентскую базу педагогов;
- Проводить мероприятия;
- В удобном виде регистрировать посещения детей на занятия;
- Планировать расписание занятий на любой период;
- Составлять график работы педагогов;
- И много другое…

ЗАКЛЮЧЕНИЕ

Результатом выполнения курсовой работы стало разработанное приложение баз данных, позволяющее автоматизировать рутинные процессы по учету и управлению данными. в детских центрах. Разработанное приложение отвечает всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации. В данной работе разработана база данных и приложение для работы совместно с базой данных.

В процессе выполнения курсовой работы были закреплены знания, полученные при изучении дисциплины. Были изучены такие пункты:

- анализ предметной области;
- построение концептуальной модели базы данных;
- организация базы данных;
- разработка прикладной программы;
- наполнение и сопровождение базы данных;

В процессе организации БД проведен до необходимого уровня абстракций анализ предметной области, построена реляционная модель БД, произведена нормализация реляционной БД. Реализация проекта была выполнена на современных программных платформах. В качестве технологии доступа к данным была использована объектно-реляционная модель, которая позволяет просто и лаконично осуществлять запросы к базе данных.

Даная курсовая работа была реализована на языке программирования C# с использованием среды разработки Microsoft Visual Studio.

Для проверки корректности работы программы были изучены методы тестирования и проведены тесты, по результатам которых были исправлены ошибки.

Цель и соответствующие задачи, поставленные перед выполнение курсового проекта выполнены в полном объёме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] В.А.Гвоздева, И.Ю.Лаврентьева. Основы построения АИС, Москва, ИД «Форум» - ИНФРА-М, 2009.
[2] А.В.Рудаков, Технология разработки программных продуктов, Москва, Издательский центр «Академия», 2008
[3] Л.Г.Гагарина, Д.В.Киселев, Е.Л.Федотова, Разработка и эксплуатация АИС. Москва, ИД «Форум» - ИНФРА-М, 2009.
[4] Г.Ю.Максимович, А.Г.Романенко, О.Ф.Самойлюк. Информационные системы. Москва 2007, Федеральное агентство по образованию
[5] Диго С.М. Базы данных: проектирование и использование: Учебник. - М.: Финансы и статистика, 2005.
[6] Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных. Учебник для вузов. - М.: Корона-принт, 2004.
[7] Кузин А.В., Левонисова С.В. Базы данных: Учебник. - М.: Academia, 2010.
[8] Джон Дей, Крейг Ван Слайк, Рэймонд Фрост Базы данных. Проектирование и разработка: Учебник. - М.: НТ Пресс, 2007.
[9] Домбровская Г.Р., Новиков Б.А. Настройка приложений баз данных: Учебник - М.: BHV, 2006.
[10] Туманов В.Е. Основы проектирования реляционных баз данных: Учебное пособие. - М.: Интернет-университет информационных технологий, 2010.
[11] Илюшечкин В. М. Основы использования и проектирования баз данных: Учебное пособие. - М.: Юрайт, 2010.
[12] Преснякова Г.В. Проектирование интегрированных реляционных баз данных: Учебник. - М.: КДУ, 2007.
[13] http://www.sql.ru/ - портал про язык SQL и клиент/серверные технологии.
[14] http://sql.itsoft.ru/ - интернет-справочник с примерами по языку SQL.
[15] http://www.cyberguru.ru/database/database-theory/ - статьи по теории баз данных.

ПРИЛОЖЕНИЕ А
(ОБЯЗАТЕЛЬНОЕ)
ФРАГМЕНТ ИСХОДНОГО КОДА

```
using MaterialSkin;
using MaterialSkin.Controls;
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace ChildCentre
{
    public partial class Main : MaterialForm
    {
        User user = new User();

        public Main()
        {
            InitializeComponent();

            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
            materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;
            materialSkinManager.ColorScheme = new ColorScheme(Primary.BlueGrey800,
Primary.BlueGrey900, Primary.BlueGrey500, Accent.LightBlue200, TextShade.WHITE);

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button3_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void button1_Click(object sender, EventArgs e)
        {


            Tables table = new Tables();
            table.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Admin admin = new Admin();
            admin.Show();
            this.Hide();
        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
```

```
        }

        private void materialRaisedButton1_Click(object sender, EventArgs e)
        {
            List<User> users = User.getUsers();
            var login = this.materialSingleLineTextField1.Text;
            var pass = this.materialSingleLineTextField2.Text;
            foreach (User item in users)
            {
                if (item.name == login && item.password == pass)
                {
                    item.updateLastSign();
                    this.user = item;
                    if (item.isAdmin == true)
                    {
                        Admin admin = new Admin();
                        admin.user = item;
                        admin.Show();
                        this.Hide();
                    } else
                    {
                        Tables tables = new Tables();
                        tables.user = item;
                        tables.Show();
                        this.Hide();
                    }
                    return;
                }
            }
            MessageBox.Show("Wrong login or password");
        }

        private void materialSingleLineTextField2_Click(object sender, EventArgs e)
        {

        }
    }
}
using MaterialSkin;
using MaterialSkin.Controls;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace ChildCentre
{
    public partial class Admin : MaterialForm
    {
        Int32 categoryCounter = 0;
        Int32 eventCounter = 0;
        Int32 childCounter = 0;
        Int32 childInfoCounter = 0;
        Int32 classCounter = 0;
        Int32 groupCounter = 0;
        Int32 listCounter = 0;
        Int32 parentsCounter = 0;
        Int32 personalCounter = 0;
        public User user;


        public Admin()
        {
```

```
            InitializeComponent();

            var materialSkinManager = MaterialSkinManager.Instance;
            materialSkinManager.AddFormToManage(this);
            materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;
            materialSkinManager.ColorScheme = new ColorScheme(Primary.BlueGrey800,
Primary.BlueGrey900, Primary.BlueGrey500, Accent.LightBlue200, TextShade.WHITE);

        }

        private void tabControl1_DrawItem(object sender, DrawItemEventArgs e)
        {

        }


        private void tabControl1_SelectedIndexChanged(object sender, EventArgs e)
        {
            TabPage tab = (sender as TabControl).SelectedTab;
            Main main = new Main();
            Console.WriteLine(tab.Name);
            if (tab.Name == "close")
            {
                this.Hide();
                user.updateLastLogout();
                main.Show();
            }
        }

        private void dataGridView3_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void Admin_Load(object sender, EventArgs e)
        {
            SetupEventTable();
            SetupCategoryTable();
            SetupChildTable();
            SetupChildInfoTable();
            SetupParentsTable();
            SetupPersonalTable();
            SetupChildTable();
            SetupClassTable();
            SetupGroupTable();
            SetupListTable();
        }

        // CATEGORY
        private void SetupCategoryTable()
        {
            this.dataGridView4.Rows.Clear();
            List<Category> categories = Category.getCategories();

            foreach (Category item in categories)
            {
                this.dataGridView4.Rows.Add(item.id, item.name, item.description, "X");
            }

            categoryCounter = 0;

        }
```

```
        private void saveCategory(object sender, EventArgs e)
        {
            DataGridView categoriesTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, categoryCounter))
            {
                var row = dataGridView4.Rows[dataGridView4.RowCount - i - 1].Cells;
                Category newCategory = new Category();
                if (row[0].Value != null)
                    newCategory.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
                    newCategory.name = row[1].Value.ToString();
                if (row[2].Value != null)
                    newCategory.description = row[1].Value.ToString();
                newCategory.AddInDB();
            }
            SetupCategoryTable();
        }

        private void dataGridView4_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView4.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
            {
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView4.Rows[e.RowIndex].Cells[0].Value.ToString();
                    Category.RemoveFromID(id);
                }
            }

        }

        private void dataGridView4_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            categoryCounter++;
        }

        // CHILD
        private void SetupChildTable()
        {
            this.dataGridView2.Rows.Clear();
            List<Child> childs = Child.getChild();

            foreach (Child item in childs)
            {
                this.dataGridView2.Rows.Add(item.id, item.id_child_info, item.id_parents,
"X");
            }

            childCounter = 0;

        }
        private void saveChild(object sender, EventArgs e)
        {
            DataGridView childsTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, childCounter))
            {
                var row = dataGridView2.Rows[dataGridView2.RowCount - i - 1].Cells;
                Child newChild = new Child();
                if (row[0].Value != null)
                    newChild.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
```

```
                newChild.id_child_info = Convert.ToInt32(row[1].Value);
            if (row[2].Value != null)
                newChild.id_parents = Convert.ToInt32(row[2].Value);
            newChild.AddInDB();
        }
        SetupChildTable();
    }

    private void dataGridView2_CellClick(object sender, DataGridViewCellEventArgs e)
    {
        if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
        var cell = dataGridView2.Rows[e.RowIndex].Cells[e.ColumnIndex];
        if (cell.Value != null)
        {
            if (cell.Value.ToString() == "X")
            {
                var id = dataGridView2.Rows[e.RowIndex].Cells[0].Value.ToString();
                Child.RemoveFromID(id);
            }
        }
    }

    private void dataGridView2_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
    {
        childCounter++;
    }

    // CHILD INFO
    private void SetupChildInfoTable()
    {
        this.dataGridView5.Rows.Clear();
        List<ChildInfo> childInfos = ChildInfo.getChildInfos();

        foreach (ChildInfo item in childInfos)
        {
            this.dataGridView5.Rows.Add(item.id, item.fio, item.dob, item.address,
"X");
        }

        childInfoCounter = 0;
    }

    private void saveChildInfo(object sender, EventArgs e)
    {
        DataGridView childsInfoTable = sender as DataGridView;
        foreach (Int32 i in Enumerable.Range(1, childInfoCounter))
        {
            var row = dataGridView5.Rows[dataGridView5.RowCount - i - 1].Cells;
            ChildInfo newChildInfo = new ChildInfo();
            if (row[0].Value != null)
                newChildInfo.id = Convert.ToInt32(row[0].Value);
            if (row[1].Value != null)
                newChildInfo.fio = row[1].Value.ToString();
            if (row[2].Value != null)
                newChildInfo.dob = Convert.ToDateTime(row[2].Value);
            if (row[3].Value != null)
                newChildInfo.address = row[3].Value.ToString();
            newChildInfo.AddInDB();
        }
        SetupChildInfoTable();
    }
```

```
        private void dataGridView5_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void dataGridView5_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView5.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
            {
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView5.Rows[e.RowIndex].Cells[0].Value.ToString();
                    ChildInfo.RemoveFromID(id);
                }
            }
        }

        private void dataGridView5_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            childInfoCounter++;
        }

        // EVENT
        private void SetupEventTable()
        {
            this.dataGridView3.Rows.Clear();
            List<Event> events = Event.getEvents();

            foreach (Event item in events)
            {
                this.dataGridView3.Rows.Add(item.id, item.description, item.date,
item.note, item.price, "X");
            }

            eventCounter = 0;
        }

        private void saveEvent(object sender, EventArgs e)
        {
            DataGridView eventsTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, eventCounter))
            {
                var row = dataGridView3.Rows[dataGridView3.RowCount - i - 1].Cells;
                Event newEvent = new Event();
                if (row[0].Value != null)
                    newEvent.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
                    newEvent.description = row[1].Value.ToString();
                if (row[2].Value != null)
                    newEvent.date = Convert.ToDateTime(row[2].Value);
                if (row[3].Value != null)
                    newEvent.note = row[3].Value.ToString();
                if (row[4].Value != null)
                    newEvent.price = Convert.ToInt32(row[4].Value);
                newEvent.AddInDB();
            }
            SetupEventTable();
        }

        private void dataGridView3_CellClick(object sender, DataGridViewCellEventArgs e)
```

```
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView3.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
            {
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView4.Rows[e.RowIndex].Cells[0].Value.ToString();
                    Event.RemoveFromID(id);
                }
            }

        }

        private void dataGridView3_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            eventCounter++;
        }

        // GROUP
        private void SetupGroupTable()
        {
            this.dataGridView1.Rows.Clear();
            List<Group> groups = Group.getGroups();

            foreach (Group item in groups)
            {
                this.dataGridView1.Rows.Add(item.id, item.id_personal, item.id_class,
item.shedule, item.size, item.price, "X");
            }

            groupCounter = 0;
        }

        private void saveGroup(object sender, EventArgs e)
        {
            DataGridView groupsTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, groupCounter))
            {
                var row = dataGridView1.Rows[dataGridView1.RowCount - i - 1].Cells;
                Group newGroup = new Group();
                if (row[0].Value != null)
                    newGroup.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
                    newGroup.id_personal = Convert.ToInt32(row[1].Value);
                if (row[2].Value != null)
                    newGroup.id_class = Convert.ToInt32(row[2].Value);
                if (row[3].Value != null)
                    newGroup.shedule = row[3].Value.ToString();
                if (row[4].Value != null)
                    newGroup.size = Convert.ToInt32(row[4].Value);
                if (row[5].Value != null)
                    newGroup.price = Convert.ToInt32(row[5].Value);
                newGroup.AddInDB();
            }
            SetupGroupTable();
        }

        private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
```

```
            {
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
                    Group.RemoveFromID(id);
                }
            }
        }
        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void dataGridView1_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            groupCounter++;
        }

        // CLASS
        private void SetupClassTable()
        {
            this.dataGridView6.Rows.Clear();
            List<Class> classes = Class.getClass();

            foreach (Class item in classes)
            {
                this.dataGridView6.Rows.Add(item.id, item.name, item.description,
item.note, item.category_id, "X");
            }

            classCounter = 0;

        }

        private void saveClass(object sender, EventArgs e)
        {
            DataGridView classesTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, classCounter))
            {
                var row = dataGridView6.Rows[dataGridView6.RowCount - i - 1].Cells;
                Class newClass = new Class();
                if (row[0].Value != null)
                    newClass.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
                    newClass.name = row[1].Value.ToString();
                if (row[2].Value != null)
                    newClass.description = row[2].Value.ToString();
                if (row[3].Value != null)
                    newClass.note = row[3].Value.ToString();
                if (row[4].Value != null)
                    newClass.category_id = row[4].Value.ToString();
                newClass.AddInDB();
            }
            SetupClassTable();
        }

        private void dataGridView6_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView6.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
            {
```

```
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView6.Rows[e.RowIndex].Cells[0].Value.ToString();
                    Class.RemoveFromID(id);
                }
            }
        }

        private void dataGridView6_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            classCounter++;
        }

        // LIST
        private void SetupListTable()
        {
            this.dataGridView7.Rows.Clear();
            List<List> lists = List.getLists();

            foreach (List item in lists)
            {
                this.dataGridView7.Rows.Add(item.id, item.id_child, item.id_group,
item.id_event, "X");
            }

            listCounter = 0;

        }

        private void saveList(object sender, EventArgs e)
        {
            DataGridView listTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, listCounter))
            {
                var row = dataGridView7.Rows[dataGridView7.RowCount - i - 1].Cells;
                List newList = new List();
                if (row[0].Value != null)
                    newList.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
                    newList.id_child = Convert.ToInt32(row[1].Value);
                if (row[2].Value != null)
                    newList.id_group = Convert.ToInt32(row[2].Value);
                if (row[3].Value != null)
                    newList.id_event = Convert.ToInt32(row[3].Value);
                newList.AddInDB();
            }
            SetupListTable();
        }

        private void dataGridView7_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView7.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
            {
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView7.Rows[e.RowIndex].Cells[0].Value.ToString();
                    List.RemoveFromID(id);
                }
            }
        }
```

```
        private void dataGridView7_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            listCounter++;
        }

        // PARENTS
        private void SetupParentsTable()
        {
            this.dataGridView8.Rows.Clear();
            List<Parents> parents = Parents.getParents();

            foreach (Parents item in parents)
            {
                this.dataGridView8.Rows.Add(item.id, item.mother_fio, item.mother_dob,
item.father_fio, item.father_dob, "X");
            }

            parentsCounter = 0;
        }

        private void saveParents(object sender, EventArgs e)
        {
            DataGridView ParentsTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, parentsCounter))
            {
                var row = dataGridView8.Rows[dataGridView8.RowCount - i - 1].Cells;
                Parents newParents = new Parents();
                if (row[0].Value != null)
                    newParents.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
                    newParents.mother_fio = row[1].Value.ToString();
                if (row[2].Value != null)
                    newParents.mother_dob = Convert.ToDateTime(row[2].Value);
                if (row[3].Value != null)
                    newParents.father_fio = row[3].Value.ToString();
                if (row[4].Value != null)
                    newParents.father_dob = Convert.ToDateTime(row[4].Value);
                newParents.AddInDB();
            }
            SetupParentsTable();
        }

        private void dataGridView8_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView8.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
            {
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView8.Rows[e.RowIndex].Cells[0].Value.ToString();
                    Parents.RemoveFromID(id);
                }
            }
        }

        private void dataGridView8_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            parentsCounter++;
        }

        // PERSONAL
```

```
        private void SetupPersonalTable()
        {
            this.dataGridView9.Rows.Clear();
            List<Personal> personals = Personal.getPersonal();

            foreach (Personal item in personals)
            {
                this.dataGridView9.Rows.Add(item.id, item.fio, item.dob,
item.qualification, "X");
            }

            personalCounter = 0;
        }

        private void savePersonal(object sender, EventArgs e)
        {
            DataGridView PersonalTable = sender as DataGridView;
            foreach (Int32 i in Enumerable.Range(1, parentsCounter))
            {
                var row = dataGridView9.Rows[dataGridView9.RowCount - i - 1].Cells;
                Personal newPersonal = new Personal();
                if (row[0].Value != null)
                    newPersonal.id = Convert.ToInt32(row[0].Value);
                if (row[1].Value != null)
                    newPersonal.fio = row[1].Value.ToString();
                if (row[2].Value != null)
                    newPersonal.dob = Convert.ToDateTime(row[2].Value);
                if (row[3].Value != null)
                    newPersonal.qualification = row[3].Value.ToString();

                newPersonal.AddInDB();
            }
            SetupPersonalTable();
        }

        private void dataGridView9_CellClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex == -1 || e.ColumnIndex == -1) return;
            var cell = dataGridView9.Rows[e.RowIndex].Cells[e.ColumnIndex];
            if (cell.Value != null)
            {
                if (cell.Value.ToString() == "X")
                {
                    var id = dataGridView9.Rows[e.RowIndex].Cells[0].Value.ToString();
                    Personal.RemoveFromID(id);
                }
            }
        }

        private void dataGridView9_RowsAdded(object sender,
DataGridViewRowsAddedEventArgs e)
        {
            personalCounter++;
        }




        private void dataGridView3_RowValidated(object sender, DataGridViewCellEventArgs
e)
        {
        }
```

```
        private void materialListView1_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
        private void dataGridView4_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

    }
}
using System;
using System.Collections.Generic;
using System.Data;


namespace ChildCentre
{
    class Category
    {
        public long id { get; set; }
        public string name { get; set; }
        public string description { get; set; }

        public Category(long id, string name, string description)
        {
            this.id = id;
            this.name = name;
            this.description = description;
        }

        public Boolean AddInDB()
        {
            DB db = new DB();

            db.bind(new string[] { "par1", this.name, "par2", this.description });

            int created = db.nQuery("INSERT INTO `ChildCentre`.`category` (`name`,
`description`) VALUES (@par1, @par2)");

            if (created > 0)
            {
                return true;
            }

            return false;
        }

        public static Boolean RemoveFromID(string id)
        {
            DB db = new DB();

            int deleted = db.nQuery("DELETE FROM category WHERE id = @id", new string[] {
"id", id });

            if (deleted > 0)
            {
                return true;
            }

            return false;
        }
        public Category()
```

```
        {

        }

        public static List<Category> getCategories()
        {
            List<Category> list = new List<Category>();

            DB db = new DB();

            DataTable items = db.query("SELECT * FROM category");

            foreach (DataRow row in items.Rows)
            {
                object[] item = row.ItemArray;

                Category newCategory = new Category();
                newCategory.id = Convert.ToInt32(item[0]);
                newCategory.name = Convert.ToString(item[1]);
                newCategory.description = Convert.ToString(item[2]);


                list.Add(newCategory);
            }

            return list;
        }
    }

}
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ChildCentre
{
    class DB
    {

        //
        private MySqlConnection connect;

        //
        private MySqlCommand command;

        //
        private MySqlDataReader reader;

        //
        private string connectiondetails = "Server=localhost;
Database=ChildCentre;Uid=root;SslMode=none";

        //
        private bool bConnected = false;
        //

        private DataTable Table;

        //
        private int affected_rows;
```

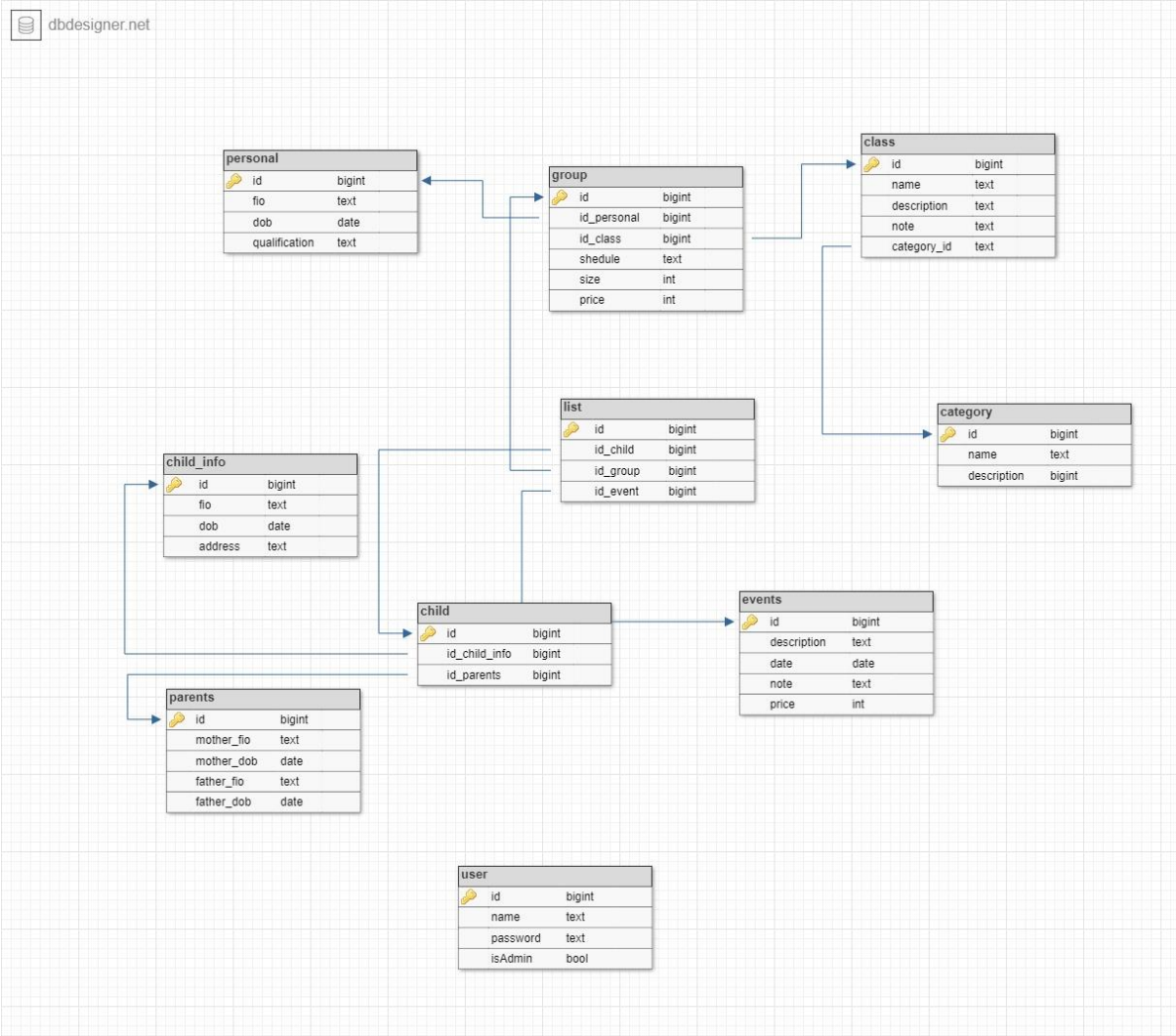
```
        //
        private string squery;

        //
        private List<string> parameters;

        public DB()
        {
            Connect();
            Table = Table = new DataTable();
            parameters = new List<string>();
        }

        private void Connect()
        {
            try
            {
                MySqlConnection connection = new MySqlConnection(connectiondetails);
                connection.Open();
                connect = connection;
                bConnected = true;
            }
            catch (MySqlException ex)
            {
                string exception = "Exception : " + ex.Message.ToString() +
"\n\rApplication will close now. \n\r" + squery;
               // MessageBox.Show(exception, "Uncaught MYSQL Exception");
                Debug(exception);

                Environment.Exit(1);
            }
        }

        public void CloseConn()
        {
            bConnected = false;
            connect.Close();
            connect.Dispose();
        }

        private void Init(string Query, string[] bindings = null)
        {
            // No connection with database? make connection
            if (bConnected == false)
            {
                Connect();
            }

            // Automatically disposes the MySQLcommand instance
            using (command = new MySqlCommand(Query, connect))
            {
                //
                bind(bindings);

                // Prevents SQL injection
                if (parameters.Count > 0)
                {
                    parameters.ForEach(delegate (string parameter)
                    {
                        string[] sparameters = parameter.ToString().Split('\x7F');
                        command.Parameters.AddWithValue(sparameters[0], sparameters[1]);
                    });
                }
```

```
                this.squery = Query.ToLower();

                if (squery.Contains("select"))
                {
                    this.Table = execDatatable();
                }
                if (squery.Contains("delete") || squery.Contains("update") ||
squery.Contains("insert"))
                {
                    this.affected_rows = execNonquery();
                }

                // Clear de parameters,
                this.parameters.Clear();
            }
        }

        public void bind(string field, string value)
        {
            parameters.Add("@" + field + "\x7F" + value);
        }

        public void bind(string[] fields)
        {
            if (fields != null)
            {
                for (int i = 0; i < fields.Length; i++)
                {
                    bind(fields[i], fields[i + 1]);
                    i += 1;
                }
            }
        }

        // Example:
        // qBind(new string[] { "12", "John" });
        // nQuery("SELECT * FROM User WHERE ID=@0 AND Name=@1");
        /// <summary>
        /// Bind multiple fields/values without identifier.
        /// </summary>
        /// <param name="fields"></param>
        public void qBind(string[] fields)
        {
            if (fields != null)
            {
                for (int i = 0; i < fields.Length; i++)
                {
                    bind(i.ToString(), fields[i]);
                }
            }
        }

        private DataTable execDatatable()
        {
            DataTable dt = new DataTable();
            try
            {
                reader = command.ExecuteReader();
                dt.Load(reader);
            }
            catch (MySqlException my)
            {
```

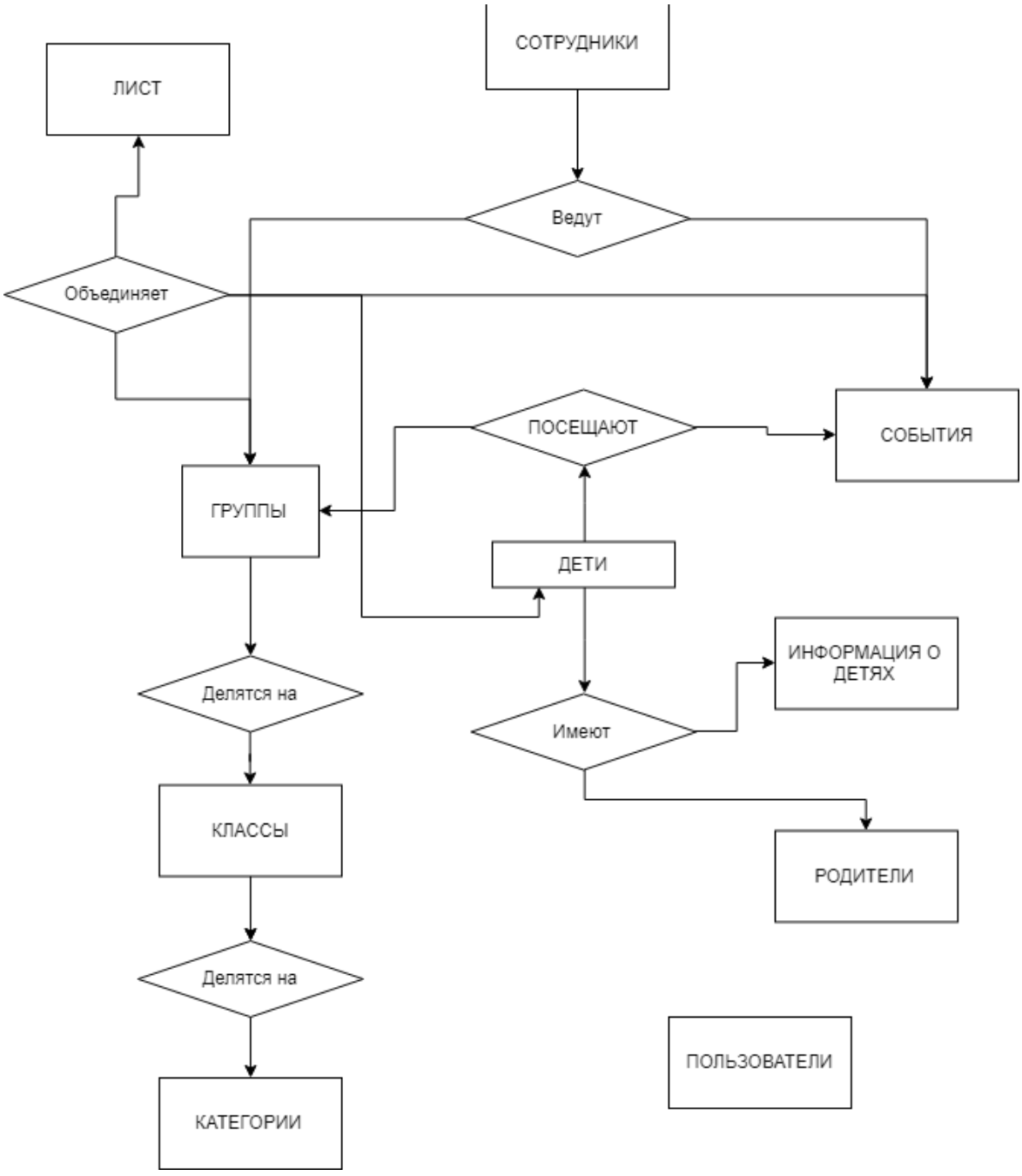
```
                string exception = "Exception : " + my.Message.ToString() + "\n\r SQL
Query : \n\r" + squery;

            //    MessageBox.Show(exception, "Uncaught MYSQL Exception");

                Debug(exception);
            }

            return dt;
        }

        private int execNonquery()
        {

            int affected = 0;

            try
            {
                affected = command.ExecuteNonQuery();
            }
            catch (MySqlException my)
            {
                string exception = "Exception : " + my.Message.ToString() + "\n\r SQL
Query : \n\r" + squery;

              //  MessageBox.Show(exception, "Uncaught MYSQL Exception");

                Debug(exception);
            }

            return affected;
        }

        public DataTable table(string table, string[] bindings = null)
        {
            Init("SELECT * FROM " + table, bindings);
            return this.Table;
        }

        public List<object> table(string table, Type t)
        {
            return new List<object>();
        }

        public DataTable query(string query, string[] bindings = null)
        {
            Init(query, bindings);
            return this.Table;
        }

        public int nQuery(string query, string[] bindings = null)
        {
            Init(query, bindings);
            return this.affected_rows;
        }

        public string single(string query, string[] bindings = null)
        {
            Init(query, bindings);

            if (Table.Rows.Count > 0)
            {
                return Table.Rows[0][0].ToString();
            }
```

```
            return string.Empty;
        }

        public string[] row(string query, string[] bindings = null)
        {
            Init(query, bindings);

            string[] row = new string[Table.Columns.Count];

            if (Table.Rows.Count > 0)
                for (int i = 0; i++ < Table.Columns.Count; row[i - 1] = Table.Rows[0][i -
1].ToString()) ;

            return row;
        }

        public List<string> column(string query, string[] bindings = null)
        {
            Init(query, bindings);

            List<string> column = new List<string>();

            for (int i = 0; i++ < Table.Rows.Count; column.Add(Table.Rows[i -
1][0].ToString())) ;

            return column;
        }


        public void Debug(string error)
        {
            Console.WriteLine(error + "/n/r");
        }
   }}
```

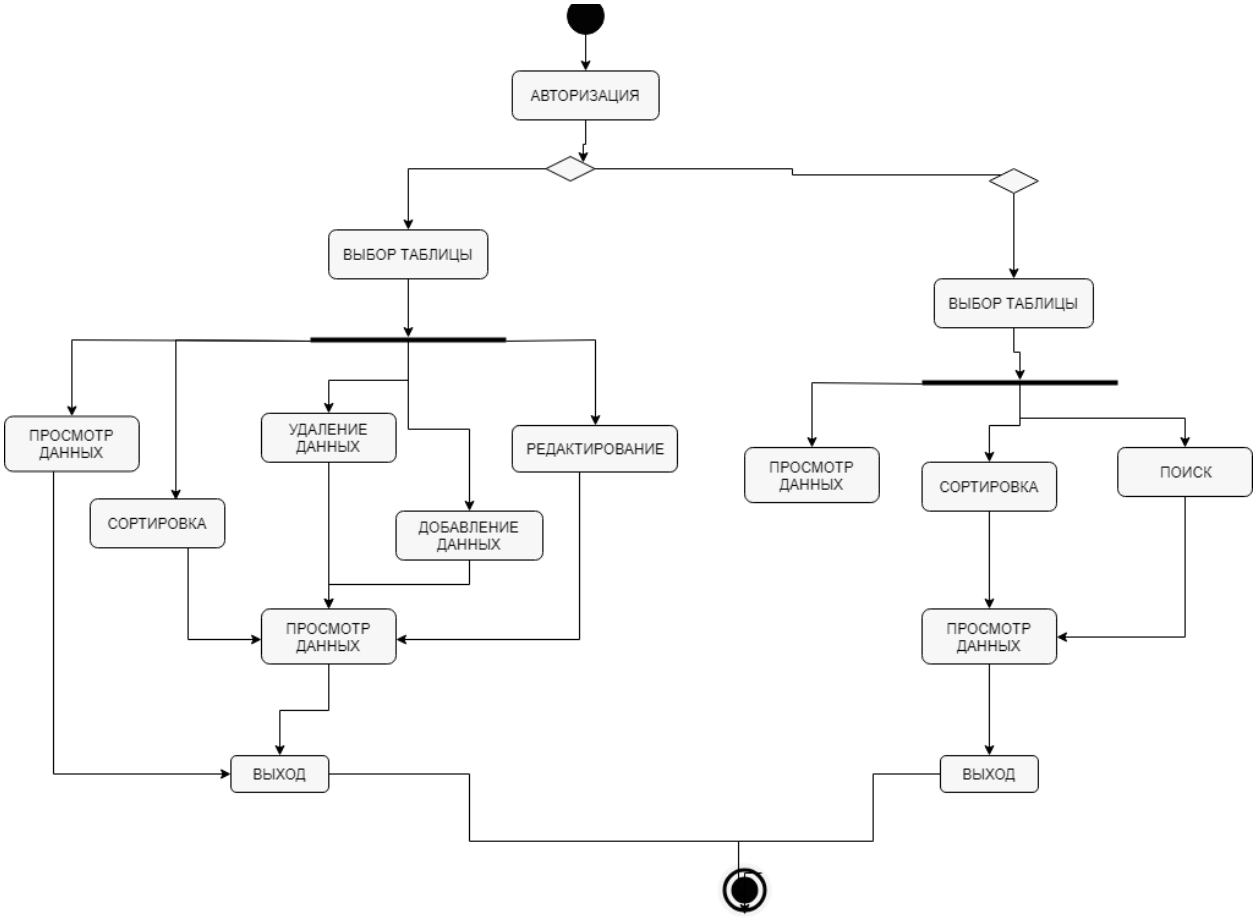
ПРИЛОЖЕНИЕ Б
(ОБЯЗАТЕЛЬНОЕ)
Схема базы



ПРИЛОЖЕНИЕ В
(ОБЯЗАТЕЛЬНОЕ)
ER-диаграмма сущностей



ПРИЛОЖЕНИЕ Г
(ОБЯЗАТЕЛЬНОЕ)
Диаграмма деятельности



ПРИЛОЖЕНИЕ Д
(ОБЯЗАТЕЛЬНОЕ)
Инфологическая модель предметной области