Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**

Кафедра компьютерных систем в управлении и проектировании (КСУП)

**К ЗАЩИТЕ ДОПУСТИТЬ**

Заведующий кафедрой КСУП

д-р техн. наук, проф.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю. А. Шурыгин

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

Бакалаврская работа по направлению подготовки  
09.03.01 «Информатика и вычислительная техника»

**ТЕМА ВКР**

**РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ ПРОВЕДЕНИЯ АНКЕТИРОВАНИЯ**

Студент гр. 588-3

\_\_\_\_\_\_\_\_\_\_\_\_\_ Емельянов Д.А.

«\_\_\_ » \_\_\_\_\_\_\_\_ 20\_\_ г.

Руководитель

Старший преподаватель

каф. КСУП

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Потапова Е. А.

«\_\_\_ » \_\_\_\_\_\_\_\_ 20\_\_ г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра компьютерных систем в управлении и проектировании (КСУП)

УТВЕРЖДАЮ

Заведующий кафедрой КСУП

д-р техн. наук, проф.

\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю. А. Шурыгин

« » 202 г.

**Индивидуальное задание**

**на выполнение выпускной квалификационной работы**

студенту Емельянову Дмитрию Алексеевичу гр. 588-3 факультета вычислительных систем

1. Тема работы: Разработка клиент-серверного приложения для проведения анкетирования   
(утверждена приказом от \_\_\_\_\_\_\_\_\_\_\_\_\_\_ № \_\_\_\_\_\_).

2. Цель работы: разработать клиент-серверное приложение для проведения анкетирования.

3. Содержание работы (перечень вопросов, подлежащих разработке):

3.1 Изучить СУБД MySQL

3.2 Изучить .Net Framework

3.3 Реализовать Базу данных MySQL на сервере

3.4 Создать клиент приложения WindowsForms

4. Дата выдачи задания: « » июня 2022 г.

5. Дата сдачи работы на кафедру: « » июля 2022 г.

Руководитель

Потапова Е.А., ст. преподаватель. \_\_\_\_\_\_\_\_\_\_\_\_ « » 202 г.

|  |  |
| --- | --- |
| (Ф.И.О., должность) (подпись) |  |
|  |  |

Задание согласовано:

|  |  |
| --- | --- |
| Задание принял к исполнению |  |
| Емельянов Д.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ « » 202 г.   |  |  | | --- | --- | | (Ф.И.О. студента) (подпись) |  | | |

**Реферат**

Бакалаврская работа 59 с., 35 рис., 16 источников.

Разработка клиент-серверного приложения для проведения анкетирования.

Объектом исследования являются система управления базами данных MySQL и платформа .NET Framework.

Цель работы: разработать клиент-серверное приложение для проведения анкетирования.

В процессе работы проводились исследования по управлению баз данных через интерфейс, созданный на базе .NET Framework.

В результате исследования бала создана система сбора данных о преподавателях посредством прохождения опросов студентами.

Пояснительная записка к бакалаврской работе выполнена в текстовом редакторе Microsoft Word 2018 и представлена на CD-диске (в конверте на обороте обложки).

**The abstract**

Bachelor's thesis 59 p., 35 pics, 16 sources.

Development of a client-server application for conducting a survey.

The object of research is the MySQL database management system and the .NET Framework platform.

The purpose of the work: to develop a client-server application for conducting a survey.

In the process of research on managing databases through an interface created on the basis of the .NET Framework.

As a result of the study of the ball, a system was created for collecting data on teachers in the course of student surveys.

An explanatory note to the bachelor's thesis is made in the text editor Microsoft Word 2018 and is presented on a CD-disk (in an envelope on the back of the cover).

# Оглавление

[Оглавление 5](#_Toc106900774)

[Введение 7](#_Toc106900775)

[1 Архитектура клиент-сервер 10](#_Toc106900776)

[1.1 Описание предметной области 10](#_Toc106900777)

[1.2 Актуальность разработки клиент-серверного приложения 12](#_Toc106900778)

[1.3 Анализ проблемы 13](#_Toc106900779)

[1.4 Требования к разрабатываемой системе 14](#_Toc106900780)

[1.5 Выбор средств разработки 15](#_Toc106900781)

[1.5.1 Платформа .NET Framework 15](#_Toc106900782)

[1.5.2 Язык программирования C# 17](#_Toc106900783)

[1.5.3 Выбор СУБД 18](#_Toc106900784)

[2 Разработка клиент-серверного приложения 21](#_Toc106900785)

[2.1 Разработка базы данных 21](#_Toc106900786)

[2.1.1 Концептуальное проектирование 21](#_Toc106900787)

[2.1.2 Логическое проектирование 23](#_Toc106900788)

[2.1.3 Физическое проектирование 23](#_Toc106900789)

[2.2 Разработка клиентской части 26](#_Toc106900790)

[2.2.1 Подключение к MySql 26](#_Toc106900791)

[2.2.2 Изучение операций с MySql 28](#_Toc106900792)

[2.2.3 Разработка интерфейса в Windows Forms 31](#_Toc106900793)

[Заключение 51](#_Toc106900794)

[Список использованных источников 52](#_Toc106900795)

# Введение

В наши дни активно развивается сфера информационных технологий, благодаря чему расширяется область её применения в человеческой деятельности. Значительную часть современных информационных технологий составляют сетевые технологии и их более глобальное проявление – Internet-технологии.

Одной из наиболее активно развивающихся областей компьютерных технологий, является автоматизация обработки информации на основе программно-вычислительных комплексов информационных систем, которые пользуются большими объемами данных, хранящимися во внешней памяти. Такие хранилища имеют сложную структуру, в основании которой лежит такой объект, как таблица. Автоматизация практически всех видов деятельности строится по принципу реляционной базы данных, что позволяет сократить финансовые и ресурсоемкие затраты, что, безусловно, является актуальной задачей в настоящее время.

Клиент-сервер – это вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемые серверами, и заказчиками услуг, называемыми клиентами.

Модель «клиент-сервер» — это структура распределенного приложения, которая разделяет задачи или рабочие нагрузки между поставщиками ресурсов или услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Часто клиенты и серверы обмениваются данными через компьютерную сеть на отдельном оборудовании, но и клиент, и сервер могут находиться в одной системе. Хост-сервер запускает одну или несколько серверных программ, которые совместно используют свои ресурсы с клиентами. Клиент обычно не делится какими-либо своими ресурсами, но запрашивает контент или услуги с сервера. Таким образом, клиенты инициируют сеансы связи с серверами, которые ожидают поступления запросов.

Существуют различные примеры клиент-серверной архитектуры в повседневной жизни. Ниже приведены некоторые примеры клиент-серверной архитектуры:

* Веб-серверы – могут размещать различные веб-сайты. Это высокопроизводительный компьютер, на котором размещены веб-сайты.
* Клиент — это пользователь, а веб-сервер — это серверная система, на которой размещены веб-сайты. Клиент или пользователь запрашивает ресурсы или данные веб-сайта сервера через высокоскоростной интернет;
* Почтовые серверы – используются для отправки и получения всех электронных писем. Различные программы позволяют обрабатывать электронную почту;
* Файловые серверы – это централизованное место для всех файлов. Например, к облачным сервисам для документов Google и Microsoft Office можно получить доступ со всех ваших устройств, а к файлам, которые вы сохраняете на своем компьютере, можно получить доступ с телефона. Таким образом, хранилища файлов централизованы и могут быть доступны нескольким клиентам.

 Клиент-серверная информационная система состоит из трех основных компонентов:

* Сервер баз данных, управляющий хранением данных, доступом и защитой, резервным копированием, отслеживающий целостность данных и выполняющий запросы клиента;
* Клиент, выполняющий логику приложения, посылающий запросы к серверу и получающий ответы от него;
* Сеть и коммуникационное программное обеспечение, которое осуществляет взаимодействие между клиентом и сервером посредством сетевых протоколов.

Рассматривая вопрос о том, какие преимущества имеет архитектура клиент-сервер, стоит сказать, что создание таких приложений позволяет добиться ряда важнейших факторов:

* Сохранность информации;
* Устойчивость к сбоям;
* Масштабируемость;
* Большая защищенность информации от несанкционированного доступа.

Недостатки клиент-серверной архитектуры

* Клиентские системы могут получить вирус или любые вредоносные скрипты, если они запущены на сервере;
* Необходимо добавить дополнительную защиту, чтобы данные не подделывались между передачей;
* Основная проблема может заключаться в том, что сервер не работает. Когда сервер не работает, клиент теряет соединение и не может получить доступ к данным.

Цель работы – разработка клиент-серверного приложения для проведения анкетирования.

Задача работы – изучение современных технологий разработки клиент-серверного приложения.

Разработанное клиент-серверное приложение предоставляет возможность общения между пользователями, сообщения которых хранятся в базе данных. Данный проект позволяет ознакомится с технологией интернет-мессенджера.

# 1 Архитектура клиент-сервер

## 1.1 Описание предметной области

Эффективность функционирования информационной системы во многом зависит от ее архитектуры.

В настоящее время перспективной является архитектура клиент-сервер.

Система клиент-сервер – это гармоничная композиция трех отдельных технологий, работающих в неразрывной связке, чтобы обеспечить эффективное хранение и быстрый доступ к данным. Рассмотрим более подробно о взаимосвязанных компонентов, данная система представлена на рисунке 1.1.

Программное обеспечение на компьютере клиенте, так называемое фронтальное ПО, отвечает за экран и ввод-вывод информации пользователем. Программное обеспечение на сервере несет ответственность за обработку введенной информации и доступ к дискам данных. К примеру, пользователь на машине-клиенте создает запрос данных в базе, фронтальная программа посылает данный запрос через сеть к серверу. Сервер базы данных проводит поиск, и отправляет обратно соответствующие запросу данные.

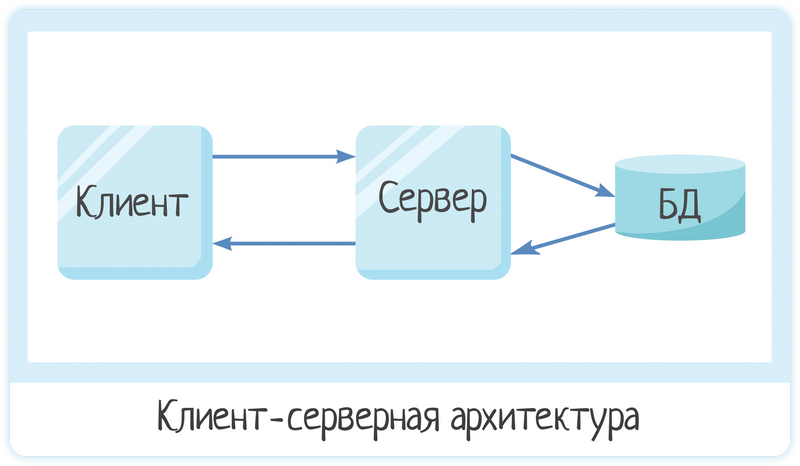


Рисунок 1.1 – Система клиент – сервер

Компьютерная сеть – это множество компьютеров, соединенных линиями связи и работающих под управлением специального программного обеспечения. Под линией связи обычно понимают совокупность технических устройств, и физической среды, обеспечивающих передачу сигналов от передатчика к приемнику.

Распределенная база данных -– набор логически связанных между собой разделяемых данных, которые физически распределены в некоторой компьютерной сети.

Распределенная СУБД (система управления базами данных) – программный комплекс, предназначенный для управления распределенными базами данных и позволяющий сделать информацию прозрачной для конечного пользователя.

Сервером определенного ресурса в компьютерной сети называется компьютер, управляющий этим ресурсом, клиентом - компьютер (программа), использующий этот ресурс. В качестве ресурса компьютерной сети могут выступать: базы данных, файловые системы, службы печати, почтовые службы. Тип сервера определяется видом ресурса, которым он управляет. Например, если управляемым ресурсом является база данных, то соответствующий сервер называется сервером базы данных.

Структура распределенной информационной системы, которая построена по архитектуре клиент-сервер с использованием сервера баз данных, обеспечивает выполнение основного объема обработки данных. Формируемые пользователем или приложением запросы поступают к серверу базы данных в виде инструкции языка SQL. Сервер базы данных выполняет поиск и извлечение нужных данных, которые затем передаются на компьютер пользователя.

Важнейшим достоинством применения базы данных в информационной системе является обеспечение независимости данных от прикладных программ, это дает возможность пользователям не заниматься проблемами представления данных на физическом уровне: размещение данных в памяти, методов доступа к ним.

Такая независимость достигается поддерживаемым СУБД многоуровневым представлением информации в базе данных на логическом (пользовательском) и физическом уровнях. Благодаря системе управления базами данных и наличию логического уровня его представления обеспечивается отделение понятийной модели базы данных от ее физического представления в памяти ЭВМ. Важнейшим параметром крупной информационной системы является быстродействие при значительном количестве пользователей, а также надежность, масштабируемость и безопасность. Всё это обеспечивает архитектура клиент-сервер.

Представленная архитектура позволяет оптимально распределить работу между клиентскими и серверной частями системы. Приложение, работающее на рабочей станции, не читает записи базы данных напрямую, а посылает запросы на сервер, где они принимаются и последовательно отрабатываются специальными программами. В результате на рабочую станцию поступают только обработанные данные, что радикально сокращает информационные потоки в локально вычислительную сеть.

## 1.2 Актуальность разработки клиент-серверного приложения

На сегодняшний день технология клиент-сервер получает все большее распространение. Она дает общее представление о том, как должна быть организована современная распределенная информационная система, и в то же время реализации этой технологии в конкретных программных продуктах и даже в видах программного обеспечения различаются весьма существенно.

Одним из важнейших преимуществ клиент-серверных информационных систем является снижение сетевого трафика при выполнении запросов.

Вторым преимуществом архитектуры клиент-сервер является возможность хранения бизнес-правил на сервере, что позволяет избежать дублирования кода в различных приложениях, использующих общую базу данных. Кроме того, в этом случае любое редактирование данных, в том числе и редактирование нештатными средствами, может быть произведено только в рамках этих правил.

Помимо перечисленных возможностей, современные серверные СУБД обладают широкими возможностями управления пользовательскими привилегиями и правами доступа к различным объектам базы данных, резервного копирования и архивации данных, а нередко и оптимизации выполнения запросов. Они также, как правило, предоставляют возможность параллельной обработки данных, особенно в случае использования многопроцессорных компьютеров в качестве сервера баз данных.

## 1.3 Анализ проблемы

Система оценки персонала – система способов измерения стоимости основного человеческого ресурса. Экономически она характеризует эффективность ресурса с точки зрения целевой функции (критерия оптимальности). Благодаря ее использованию достигается наилучшее использование имеющихся ресурсов.

Система оценки персонала включает в себя следующие виды оценки:

Аттестация объекта (работника) – процедура определения ценности (цены) направленная на подведение итогов деятельности объекта объединенная с анализом возможности достижения им плановых показателей в будущем.

Оценка объекта (работника) – проведение контрольных мероприятий направленных на контроль деятельности объекта, которые заключаются в том, чтобы убедиться, что были должным образом, выбраны и применены методы управленческого воздействия на объект.

Если раньше в основном работодатели пытались оценить одну из функций (лояльность, профессионализм и т.д.), то сейчас все больше и больше руководителей интересует, как оценить совокупный человеческий ресурс предприятия. При этом оценка человеческих ресурсов иногда заменяется оценкой кадрового потенциала (как составляющей стоимости предприятия). Это не совсем верно, так как для оценки стоимости компании важно не то, какую прибыль она может получить в будущем, а то, что она имеет сегодня и сейчас. Именно поэтому

оценка человеческого ресурса так важна для оценки стоимости активов предприятия.

Метод анкетирования – один из наиболее распространенных методов оценки. Оценочная анкета состоит из определенного набора вопросов и описаний. Оценивающий анализирует наличие или отсутствие указанных черт у оцениваемого, отмечая подходящий вариант.

В связи с расширением работ по совершенствованию системы управления высшей школой путем создания и внедрения в ВУЗах различных автоматизированных систем возникла потребность в разработке унифицированной информационной системы автоматического анкетирования на вычислительной технике. Для этого необходимо четко формализовать требования к анкетированию и разработать соответствующее алгоритмическое обеспечение.

## 1.4 Требования к разрабатываемой системе

Одним из требований, предъявляемых к разрабатываемой системе, является создание удобного, не перегруженного лишней информацией и функционалом пользовательского интерфейса, обеспечивающего легкое восприятие и обработку информации, предоставляющего свободный вывод информации в более привычном для работников предприятия виде, как например, в формате Word документа.

Помимо этого, информационная система должна обеспечивать разграничение предоставляемой информации в зависимости от роли пользователя, обладать устойчивостью к ошибкам и к негативным входным данным.

Проведем некоторые математические вычисления. Все используемые при подсчетах данные основаны на примерных показателях ТУСУР на текущий момент. Число обучающихся по очной форме обучения, принятые в 2021 году – 1598 человек, в 2020 году – 1400, в 2019 году – 1304, в 2018 году – 1321, в итоге количество студентов ТУСУР примерно равно 5623, преподавательский состав насчитывает порядка 970 человек, анкета «Преподаватель глазами студента» состоит из 10 вопросов. Учитывая, что в течение года каждый студент изучает не более 20 дисциплин, и даже если все дисциплины ведутся разными преподавателями, мы получаем, что БД будет насчитывать 4,5 миллиона записей. Однако даже это число является завышенным и в реальных условиях число записей не станет превышать и двух миллионов. Для современных систем управления базами данных эта цифра не является большой, а вычислительной мощности современных ЭВМ более чем достаточно для их обработки.

## 1.5 Выбор средств разработки

### 1.5.1 Платформа .NET Framework

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), которая подходит для различных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду. В настоящее время .NET Framework развивается в виде .NET.

Хотя .NET Framework является патентованной технологией корпорации Microsoft и официально рассчитана на работу под операционными системами семейства Windows, существуют независимые проекты (прежде всего это Mono и Portable.NET), позволяющие запускать программы .NET Framework на некоторых других операционных системах.

Основной идеей при разработке .NET Framework являлось обеспечение свободы разработчика за счёт предоставления ему возможности создавать приложения различных типов, способные выполняться на различных типах устройств и в различных средах.

Вторым принципом стала ориентация на системы, работающие под управлением семейства операционных систем Microsoft Windows

Библиотека классов платформы .NET Framework представляет собой коллекцию типов, которые тесно интегрируются со средой CLR. Библиотека классов является объектно-ориентированной. Она предоставляет типы, от которых управляемый код пользователя может наследовать функции. Это не только упрощает работу с типами .NET Framework, но и сокращает время изучения новых средств платформы .NET Framework. Кроме того, компоненты независимых производителей можно легко объединять с классами платформы .NET Framework.

Например, в классах коллекций .NET Framework реализуется набор интерфейсов для разработки пользовательских классов коллекций. Пользовательские классы коллекций легко объединяются с классами .NET Framework.

Как и ожидается от объектно-ориентированной библиотеки классов, типы .NET Framework позволяют решать типовые задачи программирования, включая работу со строками, сбор данных, подключение к базам данных и доступ к файлам. В дополнение к этим обычным задачам библиотека классов содержит типы, поддерживающие многие специализированные сценарии разработки. Можете использовать платформу .NET Framework для разработки следующих типов приложений и служб:

* Консольные приложения;
* Приложения с графическим интерфейсом Windows (Windows Forms);
* Приложения Windows Presentation Foundation (WPF);
* Приложения ASP.NET;
* службы Windows;
* Сервисноориентированные приложения, использующие Windows Communication Foundation (WCF);
* Приложения, поддерживающие бизнес-процессы Windows Workflow Foundation (WF).

Классы Windows Forms представляют собой полный набор типов, существенно упрощающих разработку графических интерфейсов пользователя Windows. При написании приложения веб-форм ASP.NET можно использовать классы веб-форм.

### 1.5.2 Язык программирования C#

C# (произносится как "си шарп") — современный объектно-ориентированный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript. Здесь представлен обзор основных компонентов языка C# 8 и более ранних версий. Если вы хотите изучить язык с помощью интерактивных примеров, рекомендуем поработать с вводными руководствами по C#.

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, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

В стандарте Ecma перечислены следующие цели разработки C#:

* Язык задуман как простой, современный, универсальный, объектно-ориентированный язык программирования;
* Язык и его реализации должны обеспечивать поддержку принципов разработки программного обеспечения, таких как строгая проверка типов, проверка границ массива, обнаружение попыток использования неинициализированных переменных и автоматическая сборка мусора. Надежность программного обеспечения, долговечность и производительность программиста важны;
* Язык предназначен для использования при разработке программных компонентов, подходящих для развертывания в распределенных средах;
* Переносимость очень важна для исходного кода и программистов, особенно для тех, кто уже знаком с C и C++;
* Поддержка интернационализации очень важна;
* C# предназначен для написания приложений как для размещенных, так и для встроенных систем, начиная от очень больших, использующих сложные операционные системы, и заканчивая очень маленькими, имеющими специальные функции;
* Хотя приложения на C# должны быть экономичными в отношении требований к памяти и вычислительной мощности, этот язык не предназначался для прямой конкуренции по производительности и размеру с C или языком ассемблера.

### 1.5.3 Выбор СУБД

MySQL - это популярный сервер баз данных, используемый в разных приложениях. SQL означает язык структурированных запросов - Structured Query Language, который MySQL использует для коммуникации с другими программами. Сверх того, MySQL имеет свои собственные расширенные функции SQL для того чтобы обеспечить пользователям дополнительный функционал. В этом документе мы рассмотрим как провести первоначальную установку MySQL, настроить базы данных и таблицы, и создать новых пользователей. Давайте начнем с установки.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

Основные преимущества MySQL:

* многопоточность, поддержка нескольких одновременных запросов;
* оптимизация связей с присоединением многих данных за один проход;
* записи фиксированной и переменной длины;
* ODBC драйвер;
* гибкая система привилегий и паролей;
* гибкая поддержка форматов чисел, строк переменной длины и меток времени;
* интерфейс с языками C и Perl, PHP;
* быстрая работа, масштабируемость;
* совместимость с ANSI SQL;
* бесплатна в большинстве случаев;
* хорошая поддержка со стороны провайдеров услуг хостинга;
* быстрая поддержка транзакций через механизм InnoDB.

Недостатки MySQL:

* MySQL не так эффективно поддерживает базы данных очень большого размера;
* MySQL не поддерживает ROLE, COMMIT и хранимые процедуры в версиях ниже 5.0;
* Сделки обрабатываются не очень эффективно;
* Есть несколько проблем со стабильностью;
* Плохо масштабируется производительность;
* Разработка не ведется сообществом, поэтому она отстает;
* Функциональность, как правило, сильно зависит от надстроек;
* Разработчики могут найти некоторые из его ограничений очень разочаровывающими.

У MySQL есть автономные клиенты, которые позволяют пользователям напрямую взаимодействовать с базой данных MySQL с помощью SQL, но чаще MySQL используется с другими программами для реализации приложений, которым требуются возможности реляционной базы данных. MySQL является компонентом программного стека веб-приложений LAMP (и других), который является аббревиатурой от Linux, Apache, MySQL, Perl/PHP/Python. MySQL используется многими веб-приложениями, управляемыми базами данных, включая Drupal, Joomla, phpBB и WordPress. MySQL также используется многими популярными веб-сайтами, включая Facebook, Flickr, MediaWiki, Twitter, и YouTube.

# 2 Разработка клиент-серверного приложения

## 2.1 Разработка базы данных

Разработка базы данных – сложный длительный процесс, который можно разделить на 3 этапа:

* Концептуальное проектирование — сбор, анализ и редактирование требований к данным;
* Логическое проектирование  — преобразование требований к данным в структуры данных;
* Физическое проектирование  — определение особенностей хранения данных, методов доступа и т. д.

Концептуальная, логическая и физическая модель — это три разных способа моделирования данных в предметной области. Хотя все они содержат сущности и отношения, они различаются целями, для которых они созданы, и целевой аудиторией. Общее понимание трех моделей заключается в том, что бизнес-аналитик использует концептуальную и логическую модель для моделирования данных, необходимых и создаваемых системой с точки зрения бизнеса, в то время как разработчик базы данных уточняет ранний проект, чтобы создать физическую модель для представления структуры физической базы данных, готовой к использованию. построение базы данных.

### 2.1.1 Концептуальное проектирование

Концептуальное проектирование  — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Концептуальная модель данных описывает, что будет содержать база данных. Разработчики баз данных создают концептуальные модели данных для организации, определения и охвата бизнес-правил и концепций, управляющих базой данных.

Концептуальное моделирование дает разработчикам возможность определить приоритеты и сотрудничать с заинтересованными сторонами в бизнесе в отношении конечного продукта.

Чаще всего концептуальная модель базы данных включает в себя:

* описание информационных объектов или понятий предметной области и связей между ними.
* описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними.

Определим все сущности предметной области. В качестве основного атрибута всех сущностей будет выступать уникальный идентификатор (далее – Id):

* Преподаватель;
* Вопросы анкеты;
* Ответы на анкеты;
* Статистика;
* Кафедра;
* Особое мнение.

Несмотря на то, что Особое мнение (далее – комментарий) является одним из вопросов анкеты, он считается не обязательным и имеет строковый тип записи, в отличии от других вопросов, поэтому выделяется как отдельная сущность.

Определим связи между сущностями:

1. На Кафедре (1) числится хотя бы один (М) Преподаватели;
2. Преподаватели (М) оцениваются по всем (–М) Вопросам анкеты;
3. Статистика (-1) высчитывается со всех (М)Ответов на анкету;
4. Каждая Анкета(1) имеет несколько (-М)Ответов на каждый вопрос;

Обозначение (–) указывает на необязательный класс принадлежности сущности. Концептуальная модель базы данных изображена на рисунке 2.1

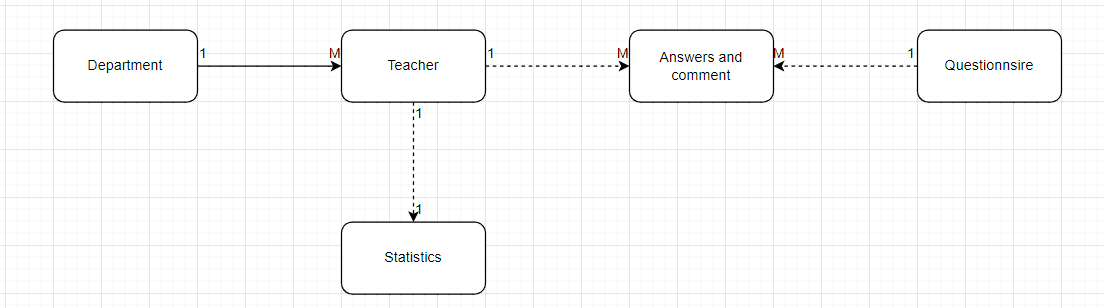


Рисунок 2.1 – Концептуальная модель базы данных

### 2.1.2 Логическое проектирование

Логическое проектирование — создание схемы базы данных на основе конкретной модели данных. Для реляционной модели данных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Логическая модель данных используется для определения структуры элементов данных и установления отношений между ними. Логическая модель данных добавляет дополнительную информацию к элементам концептуальной модели данных. Преимущество использования логической модели данных заключается в обеспечении основы для формирования базы физической модели. Однако структура моделирования остается общей.

На этом уровне моделирования данных первичный или вторичный ключ не определен. На этом уровне моделирования данных вам необходимо проверить и настроить детали соединителя, которые были установлены ранее для отношений.

Характеристики логической модели данных:

* Описывает потребности в данных для одного проекта, но может интегрироваться с другими логическими моделями данных в зависимости от масштаба проекта.
* Спроектирована и разработана независимо от СУБД.
* Атрибуты данных будут иметь типы данных с точные значения и длину.

На рисунке 2.2 изображена логическая модель базы данных, составленная по описанию отношений.

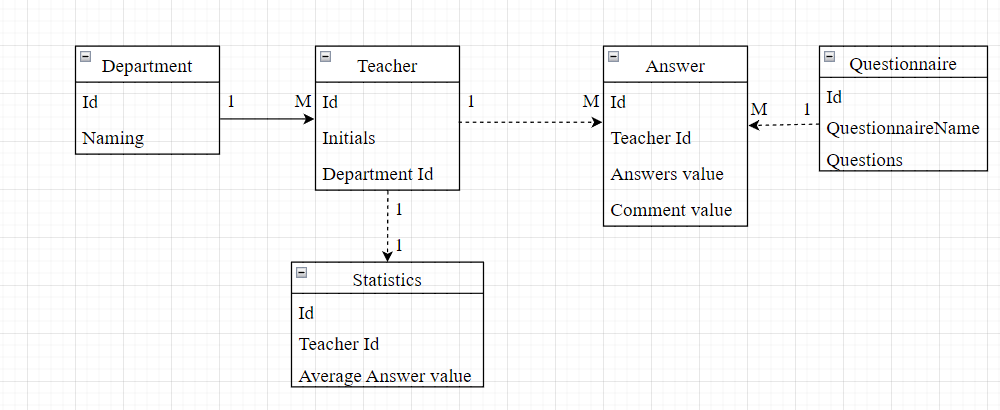


Рисунок 2.2 – Логическая модель базы данных

### 2.1.3 Физическое проектирование

**Физическое проектирование**  — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.

Физическое представление базы данных изображено на рисунке 2.3

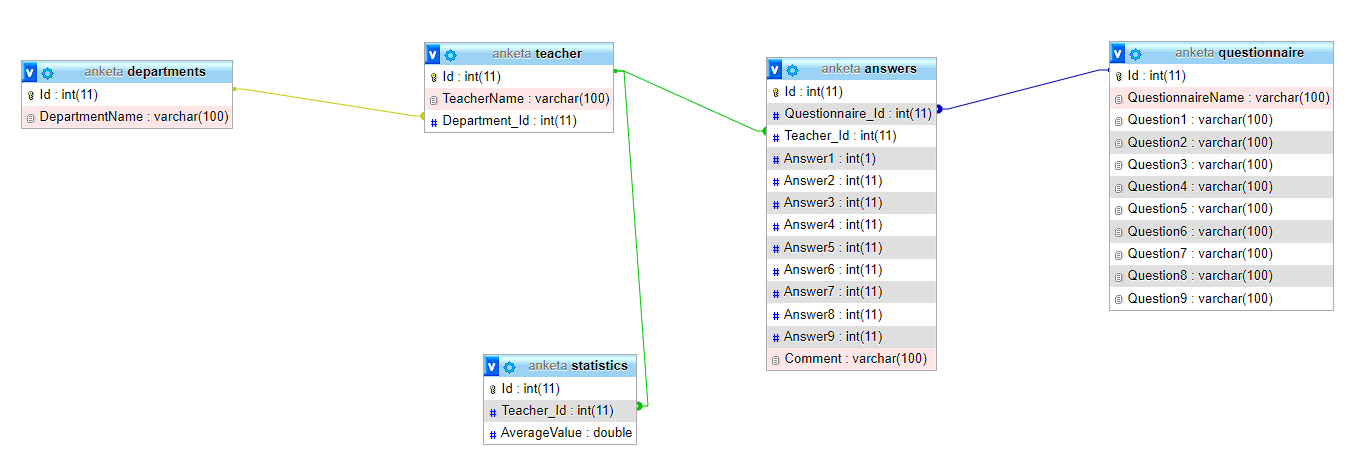


Рисунок 2.3 – Физическая модель базы данных

Описание всех таблиц базы данных приведено ниже:

1. Departments – таблица, в которой содержится список кафедр. Таблица из MySql изображена на рисунке 2.4.

Таблица состоит из Id и DepartmentName.

* Id – уникальный идентификатор;
* DepartmentName – название кафедры.

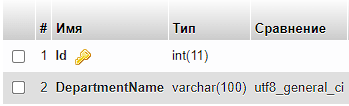


Рисунок 2.4 – Таблица Departments

1. Teacher – таблица, в которой содержится список преподавателей, каждый преподаватель принадлежит кафедре. Таблица из MySql изображена на рисунке 2.5.

Таблица состоит из Id, TeacherName, Department\_Id

* Id – уникальный идентификатор;
* TeacherName – фамилия, имя и отчество преподавателя;
* Department\_Id – уникальный идентификатор кафедры, к которой принадлежит преподаватель

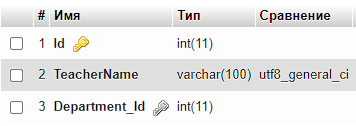


Рисунок 2.5 – Таблица Teacher

1. Questionnaire – таблица, в которой содержится список анкет и принадлежащих им вопросов. Таблица из MySql изображена на рисунке 2.6.

Таблица состоит из Id, QuestionnaireName, Question1-9.

* Id – уникальный идентификатор;
* QuestionnaireName – название анкеты;
* Question1-9 – текст вопросов;

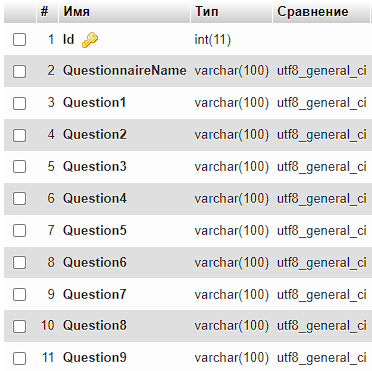


Рисунок 2.6 – Таблица Questionnaire

1. Answers – таблица, в которой содержится список ответов на вопросы. Таблица из MySql изображена на рисунке 2.7.

Таблица состоит из Id, Questionnaire\_Id, Teacher\_Id, Answer1-9, Comment.

* Id – уникальный идентификатор;
* Questionnaire\_Id – уникальный идентификатор анкеты;
* Teacher\_Id – уникальный идентификатор преподавателя, для которого проведена анкета;
* Answer1-9 – ответы на вопросы;
* Comment – комментарий к преподавателю;

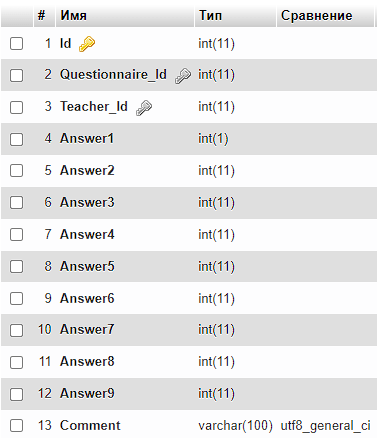


Рисунок 2.7 – Таблица Answers

1. Statistics – таблица, в которой содержится статистика всех опросов проведённых для преподавателя. Таблица из MySql изображена на рисунке 2.8.

Таблица состоит из Id, Teacher\_Id, AverageValue

* Id – уникальный идентификатор;
* Teacher\_Id – уникальный идентификатор преподавателя, по которому приведена статистика;
* AverageValue – среднее значение ответов;

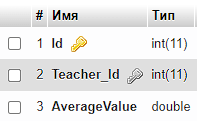


Рисунок 2.8 – Таблица Statistics

## 2.2 Разработка клиентской части

Разработка клиентской части — это тип разработки, в котором участвуют программы, работающие на клиенте или устройстве пользователя. Разработчики клиентской части сосредоточены на создании той части программы, с которой может взаимодействовать пользователь. Иногда разработку на стороне клиента также называют front-end разработкой, поскольку она фокусируется на «лицевой» части приложения, которую могут видеть пользователи. Разработчики клиента выполняют множество задач, в том числе:

* создание макетов клиента;
* разработка пользовательского интерфейса;
* добавление проверки формы;
* добавление визуальных элементов, таких как цвета и шрифты.

### 2.2.1 Подключение к MySql

Для подключения клиента к базе данных MySql для работы с ней, используется класс MySqlConnection. Для создания переменной класса MySqlConnection используется строка подключения, в которую записываются данные сервера и базы данных.

Вначале необходимо определить строку подключения, которая содержит набор параметров сервера MySql Server. Строка подключения представляет набор параметров в виде пар ключ=значение, которые отделяются друг от друга точкой с запятой.

Прежде всего, определение строки подключения зависит от типа подключения: либо подключение производится по логину и паролю, либо используется доверенное подключение (trusted connection), где не требуются логин и пароль (например, при подключении к локальному серверу SQL Server).

В случае данной программы подключение производится по логину и паролю, то общий вид строки подключения выглядит следующим образом:

"server=адрес\_сервера;port=порт;username=имя\_пользователя;password=пароль;database=название\_базы\_данных";

В случае программы “Anketa” данная строка выглядит следующим образом: "server=localhost;port=8889;username=root;password=root;database=anketa", где

Server – указывает, где размещен наш сервер, в нашем случае это localhost;

Port – порт, на котором открыт сервер базы данных так, как для создания было использовано приложение под названием MAMP, будет использован стандартный порт приложения - 8889;

Username – Имя пользователя, который подключается к базе данных, приложение MAMP использует стандартный логин - root;

Password – Пароль пользователя, приложение MAMP использует стандартный пароль - root;

Database – Название созданной базы данных MySql.

Данная строка передаётся при создании экземпляра класса MySqlConnection, синтаксис создания переменной подключения показан ниже:

MySqlConnection connection = new MySqlConnection("server=localhost;port=8889;username=root; password=root;database=anketa");

Мы всегда должны открывать соединение перед запросом нашей таблицы (таблиц) и закрывать его сразу после того, как мы закончим, чтобы освободить ресурсы и указать, что это соединение больше не нужно.

Открытие и закрытие соединения с базой данных очень просто, однако всегда лучше использовать обработку исключений перед открытием или закрытием соединения, чтобы отлавливать ошибки и устранять их.

Для открытия соединения будет использован статический метод из абстрактного класса MySql MySqlConnection OpenConnection(), синтаксис которого изображён ниже:

public static MySqlConnection OpenConnection()

{

MySqlConnection connection = new MySqlConnection

("server=localhost;port=8889;username=root;" +

"password=root;database=anketa");

if (connection.State == ConnectionState.Closed)

{

connection.Open();

}

return connection;

}

Для закрытия соединения создан метод MySqlConnection CloseConnection(), синтаксис которого приведён ниже

public static void CloseConnection(MySqlConnection connection)

{

if (connection.State == ConnectionState.Open)

{

connection.Close();

}

}

В начале каждого метода с использованием базы данных MySql будет создано соединение, после окончания взаимодействий с базой данных, соединение будет закрыто, например, метод GetDepartmentNames() будет выглядеть так:

public static string[] GetDepartmentNames()

{

//Открыть соединение

MySqlConnection connection = MySql.OpenConnection();

//Создать команду

MySqlCommand command = connection.CreateCommand();

//текст программы

//Закрыть соединение

MySql.CloseConnection(connection);

return row;

}

### 2.2.2 Изучение операций с MySql

Обычно, команды Insert, Update и Delete используются для записи, изменения и удаления записей в базе данных, а команда Select используется для чтения данных.

По этой причине у нас есть разные типы методов для выполнения этих запросов.

Методы следующие:

ExecuteNonQuery: используется для выполнения команды, которая не возвращает никаких данных, например, вставить, обновить или удалить.

ExecuteReader: используется для выполнения команды, которая вернет 0 или более записей, например Select.

Разберём команды Insert, Update и Delete, которые являются самыми простыми. Процесс успешного выполнения команды выглядит следующим образом:

1. Открыть соединение с базой данных;
2. Создать команду MySQL;
3. Назначить запрос команде. Это можно сделать с помощью конструктора или с помощью методов Connection и CommandText в классе MySqlCommand;
4. Выполнить команду;
5. Закрыть соединение.

Таким образом, метод добавления значения в таблицу будет выглядеть так:

//Метод вставки записи

public void Insert()

{

//Задаём строку записи, в которой 1st\_column и 2nd\_column это названия столбцов, в которые будут записаны значения 1st\_value и 2nd\_value соответственно

string query = "INSERT INTO table\_name (1st\_column, 2nd\_column) VALUES('1st\_value', '2nd\_value')";

//Открыть соединение

MySqlConnection connection = MySql.OpenConnection();

//создать команду и записать в неё текст запроса и соединение, по которому будет выполнена команда

MySqlCommand command = new MySqlCommand(query, connection);

//Выполнить команду

command.ExecuteNonQuery();

//Закрыть соединение

MySql.CloseConnection(connection);

}

//Метод обновления записи

public void Update()

{

//Задаём строку обновления, в которой 1st\_column и 2nd\_column это названия столбцов, в таблице table\_name будет изменено значение 1st\_column с 1st\_value на 3rd\_value, если такая запись существует

string query = "UPDATE table\_name SET 1st\_column='3rd\_value', 2nd\_column='2nd\_value' WHERE 1st\_column='1st\_value'";

//Открыть соединение

MySqlConnection connection = MySql.OpenConnection();

//создать команду и записать в неё текст запроса и соединение, по которому будет выполнена команда

MySqlCommand command = new MySqlCommand(query, connection);

// Выполнить команду

command.ExecuteNonQuery();

//Закрыть соединение

MySql.CloseConnection(connection);

}

//Метод удаления записи

public void Delete()

{

//Задаём строку удаления, в которой запись со значением 1st\_column будет удалена, если такая запись существует

string query = "DELETE FROM table\_name WHERE 1st\_column='1st\_value'";

//Открыть соединение

MySqlConnection connection = MySql.OpenConnection();

//создать команду и записать в неё текст запроса и соединение, по которому будет выполнена команда

MySqlCommand command = new MySqlCommand(query, connection);

//Выполнить команду

command.ExecuteNonQuery();

//Закрыть соединение

MySql.CloseConnection(connection);

}

Чтобы выполнить оператор Select, добавляются еще несколько шагов и используется метод ExecuteReader, который возвращает объект dataReader для чтения и вывода значений записей из таблицы.

1. Открыть соединение с базой данных;
2. Создать команду MySQL;
3. Назначить соединение и запрос команде. Это можно сделать с помощью конструктора или с помощью методов Connection и CommandText в классе MySqlCommand;
4. Создать объект MySqlDataReader для чтения выбранных записей/данных;
5. Выполнить команду;
6. Прочитать записи и отобразите их или сохраните их в списке;
7. Закрыть считыватель данных;
8. Закрыть соединение.

//Метод выбора

public List<string>[] Select()

{

string query = "SELECT \* FROM table\_name";

//Создать список для хранения результатов

List<string>[] list = new List<string>[3];

list[0] = new List<string>();

list[1] = new List<string>();

list[2] = new List<string>();

//Открыть соединение

MySqlConnection connection = MySql.OpenConnection();

//Создать команду и записать в неё текст запроса и соединение, по которому будет выполнена команда

MySqlCommand command = new MySqlCommand(query, connection);

//Создать Data Reader и выполнить команду чтения

MySqlDataReader dataReader = command.ExecuteReader();

//Прочитать данные и записать их в список

while (dataReader.Read())

{

list[0].Add(dataReader["id"]);

list[1].Add(dataReader["1st\_column"]);

list[2].Add(dataReader["2nd\_column"]);

}

//Закрыть Data Reader

dataReader.Close();

//Закрыть соединение

MySql.CloseConnection(connection);

//Вернуть список со значениями

return list;

}

### 2.2.3 Разработка интерфейса в Windows Forms

Приложение Windows Forms — это приложение, работающее на настольном компьютере. Приложение Windows Forms обычно имеет набор элементов управления, таких как метки, текстовые поля, списки и т. д.

При запуске приложения Anketa, пользователь увидит окно входа, интерфейс которого изображён на рисунке 2.9.

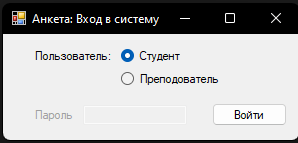


Рисунок 2.9 – Окно входа в программу

Приложение разрабатывалось с помощью интерфейса WinForms. При реализации окна входа в систему были использованы следующие компоненты:

− Button: кнопка подтверждения входа в систему;

− TextBox: текстовое поле ввода данных от пользователя;

− RadioButton: элемент, предоставляющий пользователю выбор условий;

− Label: надпись-подсказка для пользователя.

Теперь рассмотрим логику события, которое вызывается нажатием на кнопку. Для начала проверяется, какой RadioButton выбран.

При выбранной RadioButton «Студент» доступна только кнопка «Войти», при нажатии на которую, откроется окно студента.

При выбранной RadioButton «Преподаватель», станет доступен Label, в котором будет написано «Пароль» и TextBox для ввода пароля, в который необходимо ввести пароль. Данное изменение изображено на рисунке 2.10.

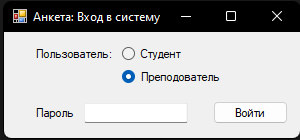


Рисунок 2.10 – Интерфейс окна при выборе RadioButton «Преподаватель»

После ввода в TextBox неверного значения и нажатия на кнопку «Войти», откроется окно ошибки, в котором будет написано, что введён неверный пароль. Данное окно изображено на рисунке 2.11.

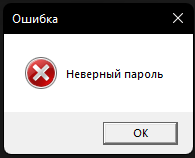


Рисунок 2.11 – Сообщение об ошибке ввода пароля

Следующее окно, которое будет открыто зависит от того, что выбрано в RadioButton. Если выбран «Студент», будет открыто окно «Анкета: Студент», а если выбран «Преподаватель», то пользователь должен написать пароль и после проверки правильности ввода пароля, откроется окно «Анкета: Администратор»

Если выбран RadioButton «Студент», откроется окно «Анкета: Студент», изображённое на рисунке 2.12.

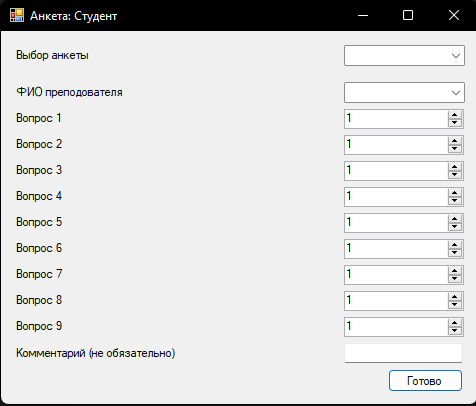


Рисунок 2.12 – Окно «Анкета: Студент»

В реализации окна «Анкета: Студент» были использованы следующие компоненты:

* ComboBox: предоставляет пользователю выбор из нескольких значений;
* NumericUpDown: поле ввода числовых значений, ограниченных с обеих сторон;
* TextBox: поле ввода комментария.

Теперь рассмотрим алгоритм поведения окна на внесённые пользователем изменения.

При нажатии на ComboBox «Выбор анкеты», пользователь увидит варианты анкет для выбора, на данный момент доступна только одна анкета, окно приложения после нажатия на ComboBox изображено на рисунке 2.13.

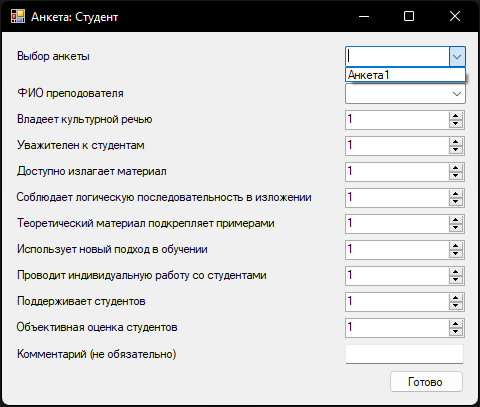


Рисунок 2.13 – Окно «Анкета: Студент»

После выбора данной анкеты, текст надписей изменится на критерии по оценке преподавателей, данное изменение изображено на рисунке 2.14.

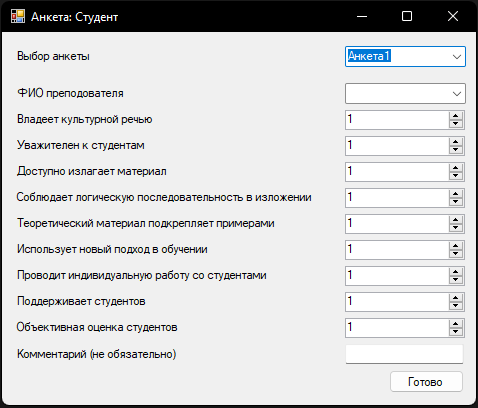


Рисунок 2.14 – Окно «Анкета: Студент» после выбора анкеты

Таким же образом можно выбрать преподавателя. После выбора преподавателя, пользователь оценивает его по всем критериям анкеты, а так же по желанию может написать комментарий об этом преподавателе. Заполненная анкета изображена на рисунке 2.15.

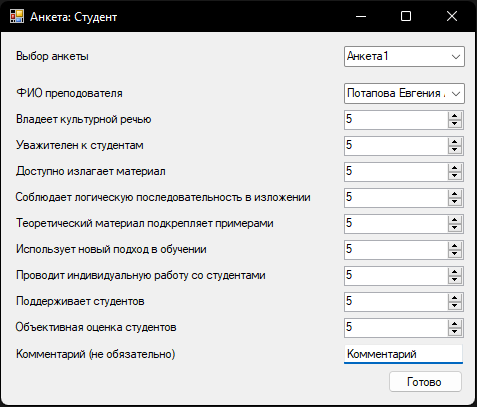


Рисунок 2.15 – Заполненное окно «Анкета: Студент»

После нажатия на кнопку «Готово», пользователь увидит сообщение, которое сигнализирует ему, что его ответ отправлен, данное сообщение изображено на рисунке 2.16.

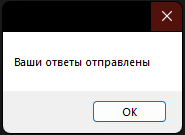


Рисунок 2.16 – Сообщение об отправке данных

Обновив таблицу MySql можно увидеть, что ответы действительно занесены в базу данных. Таблица изображена на рисунке 2.17

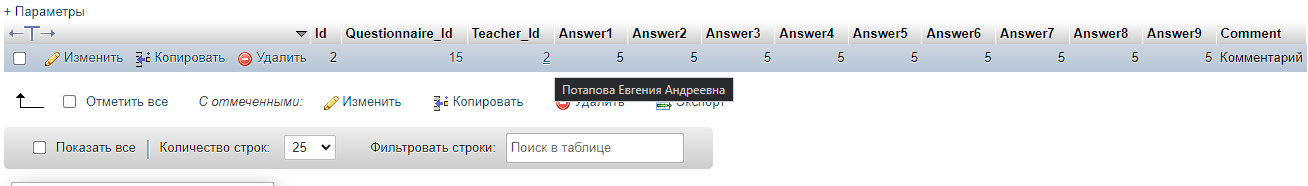


Рисунок 2.17 – Данные, занесённые в таблицу

Так же, если поля не будут заполнены, будет выведено сообщение, в котором сказано, что нужно заполнить все поля, данное сообщение изображено на рисунке 2.18

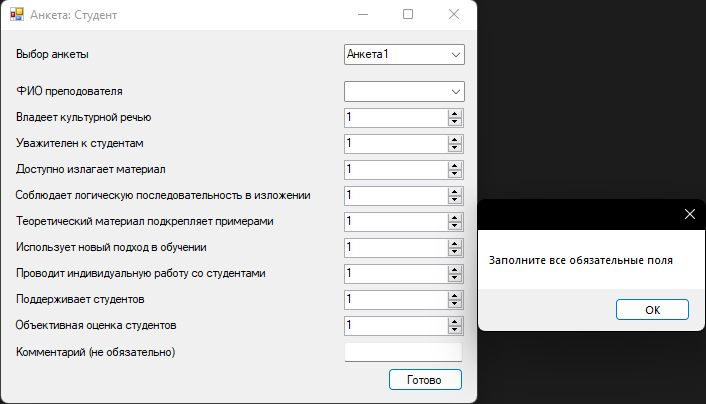


Рисунок 2.18 – Сообщение ошибки заполнения полей

Если выбран RadioButton «Преподаватель» и введён верный пароль, откроется окно «Анкета: Преподаватель», изображённое на рисунке 2.19.

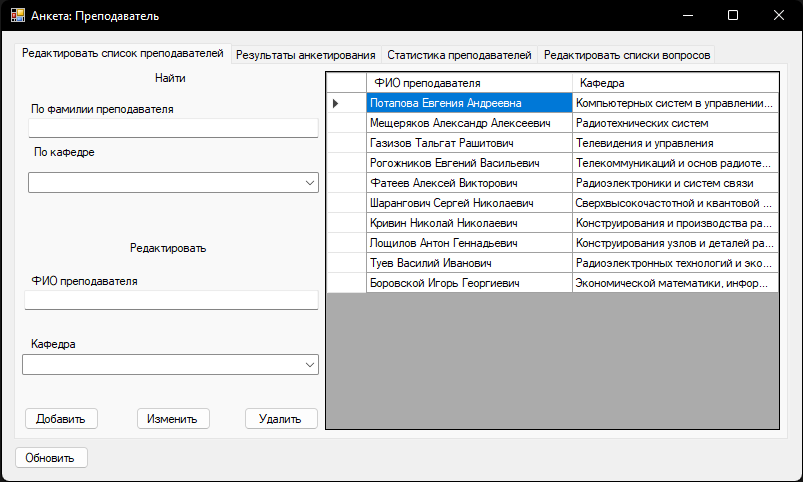


Рисунок 2.19 – Окно «Анкета: Преподаватель»

В реализации окна «Анкета: Преподаватель» были использованы следующие компоненты:

* TabControl – содержит страницы вкладок, которые представлены объектами TabPage, которые добавляются через свойство TabPages. Порядок страниц вкладок в этой коллекции отражает порядок отображения вкладок в элементе управления.

На каждой TabPage данного элемента размещены:

* TextBox – с помощью данного элемента управления пользователь может вводить текст в приложении. В данном случае поле используется для поиска и добавления нового преподавателя в таблицу.
* ComboBox – отображает TextBox в сочетании с ListBox, что позволяет пользователю выбирать элементы из списка или вводить новое значение.
* Button – производит некоторое действие, используя код программы, который записан в кнопку.
* DataGridView - предоставляет настраиваемую таблицу для отображения данных. Класс DataGridView позволяет отображать списки данных в нескольких строках и нескольких столбцах.

В TabPage1, которая называется «Редактировать список преподавателей» пользователю предоставляется возможность поиска по имени преподавателя, по кафедре, в которой он состоит, а так же добавление в базу данных, изменение и удаление выбранного в DataGridView преподавателя в базе данных. При запуске данного окна происходит обновление DataGridView, то есть информация из базы данных помещается в таблицу DataGridView.

Рассмотрим методы поиска, добавления в таблицу, изменение значений таблицы, удаление данных, а так же метод обновления DataGridView, которые были созданы для данной TabPage.

Для поиска нужных строк в базе данных используется метод под названием UpdateTeacherListDataGridView, так же данный метод полностью обновляет значения, которые изображены в таблице. Синтаксис данного метода объяснён ниже:

private void UpdateTeacherListDataGridView()

{

//Очистка строк в DataGridView

TeacherListDataGridView.Rows.Clear();

//Открытие соединения

MySqlConnection connection = MySql.OpenConnection();

//Создание команды с командной строкой MySql

MySqlCommand command = new MySqlCommand("SELECT " +

"`TeacherName`,`DepartmentName` FROM `departments`" +

",`teacher` WHERE departments.Id = teacher." +

"Department\_Id AND `TeacherName` LIKE '%" +

searchTextBox1.Text + "%' AND `DepartmentName` " +

"LIKE '%" + searchComboBox1.Text + "%'", connection);

//Создать Data Reader и выполнить команду чтения

MySqlDataReader reader = command.ExecuteReader();

//Прочитать данные и записать их в DataGridView построчно

while (reader.Read())

{

MySql.TeacherListReadSingleRow(TeacherListDataGridView

,reader);

}

//Закрыть Data Reader

reader.Close();

//Закрыть соединение

MySql.CloseConnection(connection);

}

В командной строке MySql можно увидеть, что в строку так же добавляются переменные searchTextBox1.Text и searchComboBox1.Text. С помощью данных переменных из базы данных выбираются строки, в которых TeacherName имеет что-то общее с данными, введёнными в searchTextBox1, а так же, строки, в которых DepartmentName равно данным из searchComboBox1. SearchTextBox1 и searchComboBox1 выделены на рисунке 2.20

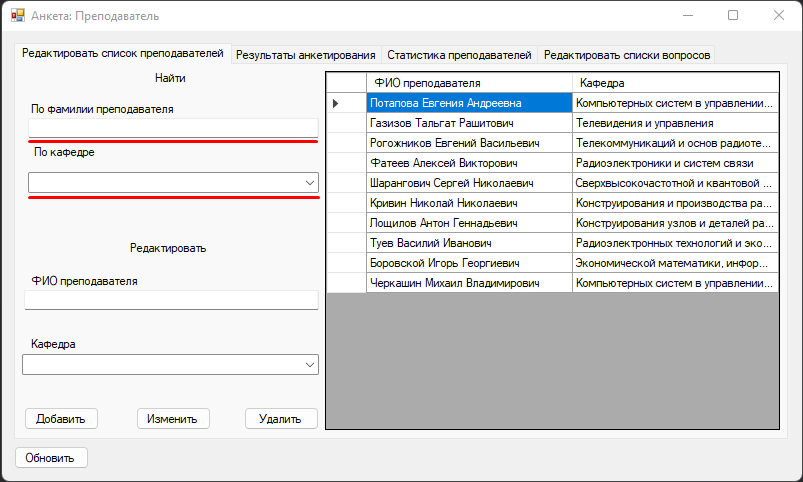


Рисунок 2.20 – Окно «Анкета: Преподаватель» с выделенными searchTextBox1 и searchComboBox1

Результаты поиска изображены на рисунке 2.21.

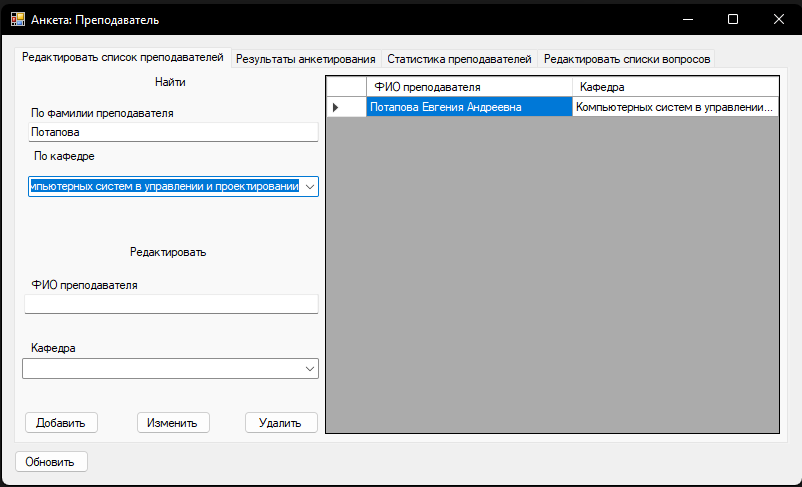


Рисунок 2.21 – Окно «Анкета: Преподаватель» с выполненным поиском

Для добавления нужных строк в базу данных используется метод под названием AddRowTeacherListDataGridView, так же, после обновления данные DataGridView полностью обновляются, которые изображены в таблице. Синтаксис данного метода объяснён ниже:

private void AddRowTeacherListDataGridView()

{

//Если TextBox и ComboBox не пусты, выполнить алгоритм, иначе вывести сообщение

if (EditComboBox.Text != string.Empty || EditTextBox.Text != string.Empty)

{

//Открытие соединения

MySqlConnection connection = MySql.OpenConnection();

//Создание команды добавления с командной строкой MySql

MySqlCommand addCommand = new MySqlCommand($"INSERT" +

$" INTO `teacher` (`TeacherName`,`Department\_Id`)" +

$" VALUES (@teacherName, @departmentId)", connection);

//Поиск Id выбранной кафедры, используя метод из вспомогательного класса

int departmentId = MySql.GetDepartmentsId(EditComboBox.Text);

//Замена параметров на значения переменных

addCommand.Parameters.Add("@teacherName",

MySqlDbType.VarChar).Value

= EditTextBox.Text;

addCommand.Parameters.Add("@departmentId",

MySqlDbType.Int32).Value

= departmentId;

//Проверка выполнения команды

if (addCommand.ExecuteNonQuery() == 1)

{

MessageBox.Show("Преподаватель был добавлен");

}

else

{

MessageBox.Show("Преподаватель не был добавлен");

}

MySql.CloseConnection(connection);

}

else

{

MessageBox.Show("Введите данные для добавления");

}

}

Результат выполнения данного метода изображён на рисунке 2.22

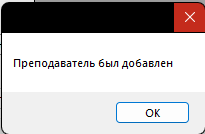


Рисунок 2.22 – Окно подтверждения добавления преподавателя

После автоматического обновления DataGridView в списке преподавателей добавилась новая запись, которая изображена в конце списка на рисунке 2.23.

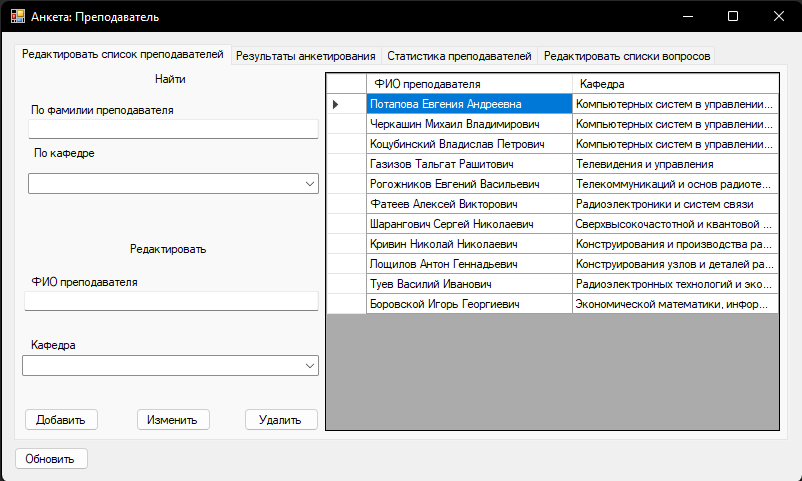


Рисунок 2.23 – Окно «Анкета: Преподаватель» с добавленной записью

Для изменения данных в базе данных пользователю требуется выбрать изменяемую запись, после чего она появится в разделе «Редактировать», результат выбора записи изображён на рисунке 2.24. Синтаксис метода изменения приведён ниже

private void EditRowTeacherListDataGridView()

{

//Проверка выбрана ли запись

int index = - 1;

index = TeacherListDataGridView.CurrentCell.RowIndex;

//Если запись выбрана, выполнить алгоритм, иначе вывести сообщение

if (index != -1)

{

//Открытие соединения

MySqlConnection connection = MySql.OpenConnection();

//Поиск Id выбранного преподавателя, используя метод из вспомогательного класса

int teacherId = 0;

teacherId = MySql.GetTeacherId(TeacherListDataGridView.

Rows[index].Cells[0].Value.ToString());

//Поиск Id выбранной кафедры, используя метод из вспомогательного класса

int departmentId = MySql.GetDepartmentsId

(EditComboBox.Text);

//Если преподаватель существует в базе данных,

//выполнить алгоритм, иначе вывести сообщение

if (teacherId != 0)

{

//Создание команды удаления ответов, связанных с данным преподавателем

MySqlCommand command = new MySqlCommand($"DELETE " +

$"FROM `answers` WHERE `Teacher\_Id` LIKE '%"

+ teacherId + "%'", connection);

command.ExecuteNonQuery();

//Создание команды удаления статистики, связанной с данным преподавателем

command = new MySqlCommand($"DELETE FROM " +

$"`statistics` WHERE " +

$"`Teacher\_Id` LIKE '%" + teacherId +

"%'", connection);

command.ExecuteNonQuery();

//Создание команды изменения данных преподавателя

command = new MySqlCommand($" UPDATE `teacher`" +

$" SET `TeacherName` = '{EditTextBox.Text}'," +

$" `Department\_Id` = '{ departmentId }'" +

$" WHERE Id LIKE '{teacherId}'", connection);

command.ExecuteNonQuery();

MessageBox.Show("Преподаватель был изменён");

}

else

{

MessageBox.Show("Преподаватель не был изменён");

}

MySql.CloseConnection(connection);

}

else

{

MessageBox.Show("Выберите преподавателя для изменения");

}

}

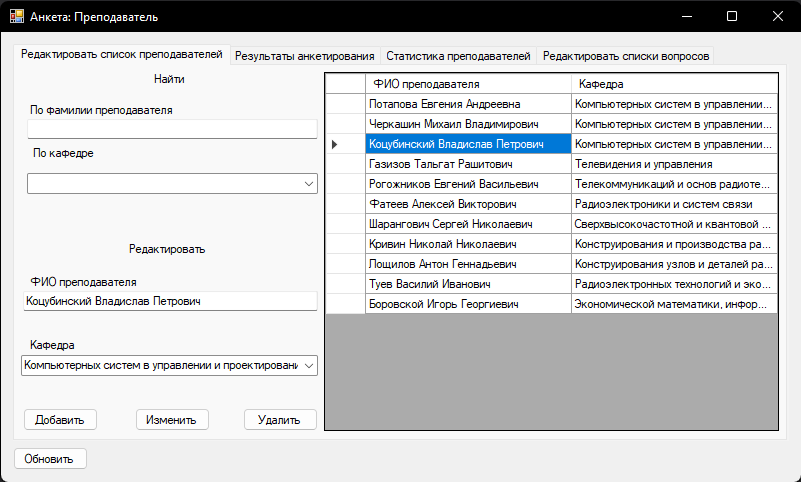


Рисунок 2.24 – Окно «Анкета: Преподаватель» с выбранной записью

После изменения, запись будет записана на том же месте, где была, но с изменёнными значениями.

Результат изменения изображён на рисунке 2.25

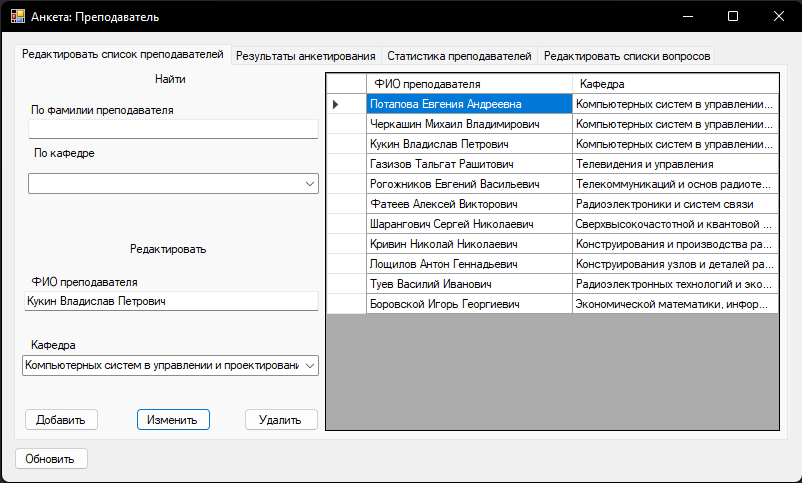


Рисунок 2.25 – Окно «Анкета: Преподаватель» с изменённой записью

Для удаления записи в базе данных пользователю требуется выбрать удаляемую запись в DataGridView и нажать на Button «удалить».

Результат выполнения изображён на рисунке 2.26.

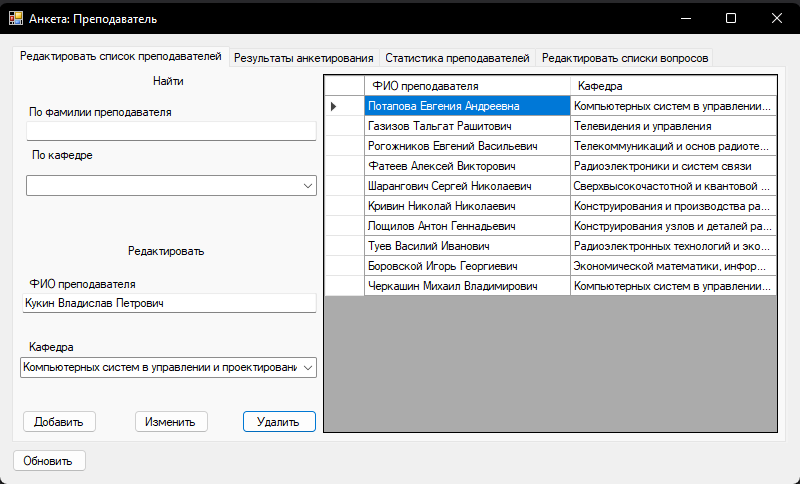


Рисунок 2.26 – Окно «Анкета: Преподаватель» после удаления записи

В TabPage2, которая называется «Результаты анкетирования» пользователю предоставляется возможность поиска по имени преподавателя, по кафедре, в которой он состоит, а так же по названию анкеты, При запуске данного окна происходит обновление DataGridView, то есть информация из базы данных помещается в таблицу DataGridView. Данная страница изображена на рисунке 2.27.

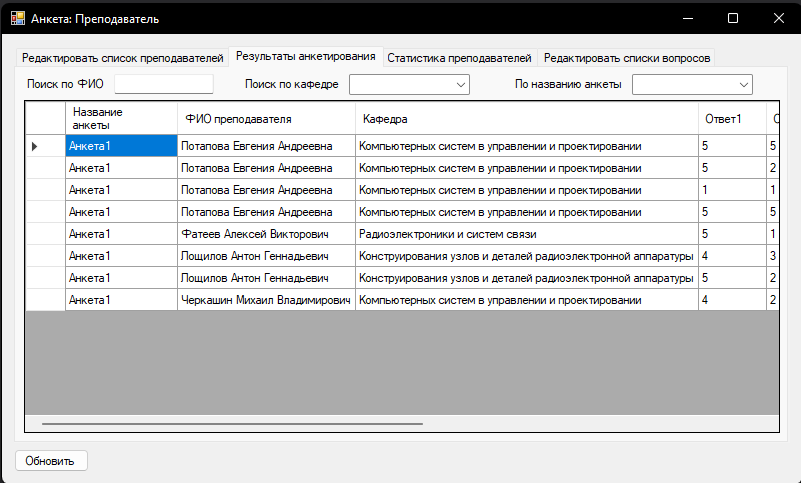


Рисунок 2.27 – Окно «Анкета: Преподаватель» с выбранной TabPage «Результаты анкетирования»

Как можно увидеть, на таблице изображено несколько записей, запись, созданная на рисунке 2.15, а так же новые. Так же данные записи можно увидеть в таблице базы данных MySql, изображённой на рисунке 2.28.



Рисунок 2.28 – Таблица answers в базе данных MySql

При выборе таблицы по названию анкеты, названия столбцов, названные «Вопрос(номер вопроса)», изменятся на вопросы данной анкеты, например, выделим результаты поиска «Анкета1». Результаты поиска изображены на рисунке 2.29

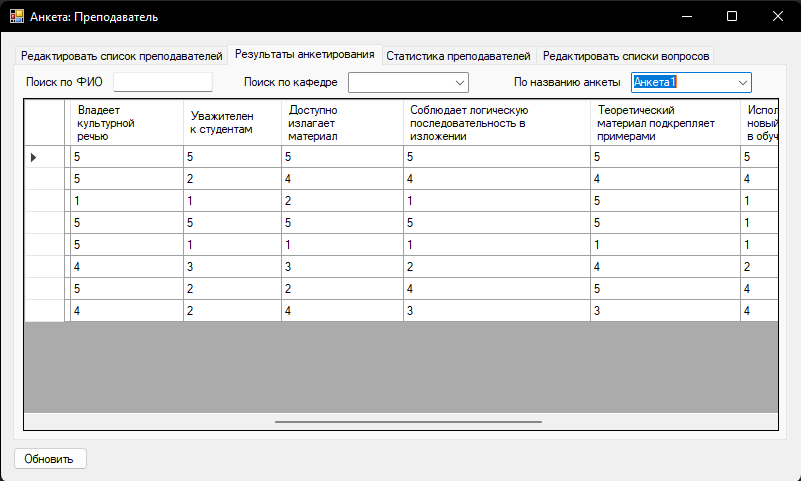


Рисунок 2.29 – TabPage «Результаты анкетирования» с поиском по анкете

В TabPage3, которая называется «Cтатистика преподавателей» пользователю предоставляется возможность просмотра среднего значения всех тестов, проведённых для каждого преподавателя, так же реализованы алгоритмы поиска по имени преподавателя и по кафедре, в которой он состоит. При запуске данного окна происходит обновление DataGridView, то есть информация из базы данных помещается в таблицу DataGridView. Данная страница изображена на рисунке 2.30.

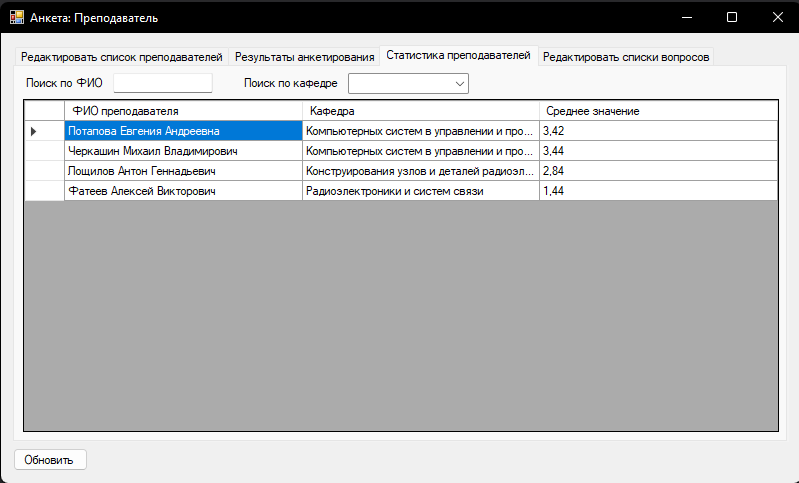


Рисунок 2.30 – TabPage «Статистика преподавателей»

Как можно увидеть, в DataGridView изображены столбцы «ФИО преподавателя», «Кафедра» и «Среднее значение». В данной таблице можно выполнить поиск по имени преподавателя, так же по названию кафедры. Синтаксис нахождения среднего значения приведён в приложении А.

В TabPage4, которая называется «Редактировать списки вопросов» пользователю предоставляется возможность добавления, изменения и удаления анкет. При запуске данного окна происходит обновление DataGridView, то есть информация из базы данных помещается в таблицу DataGridView. Данная страница изображена на рисунке 2.31.

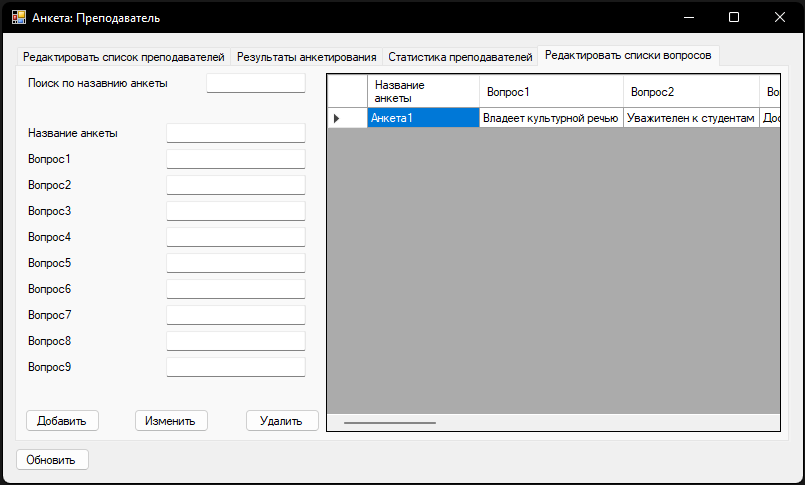


Рисунок 2.31 – TabPage «Редактировать списки вопросов»

Как можно увидеть, в DataGridView создана одна анкета с названием «Анкета1», на данную анкету и был создан ответ на рисунке 2.15.

Для добавления новой анкеты в базу данных, необходимо заполнить все TextBox и нажать на кнопку «Добавить», после чего будет проверены TextBox на заполненность, создано подключение и значения из полей ввода будут перенесены в таблицу «questionnaire» в базе данных. Результат изображён на рисунке 2.32.

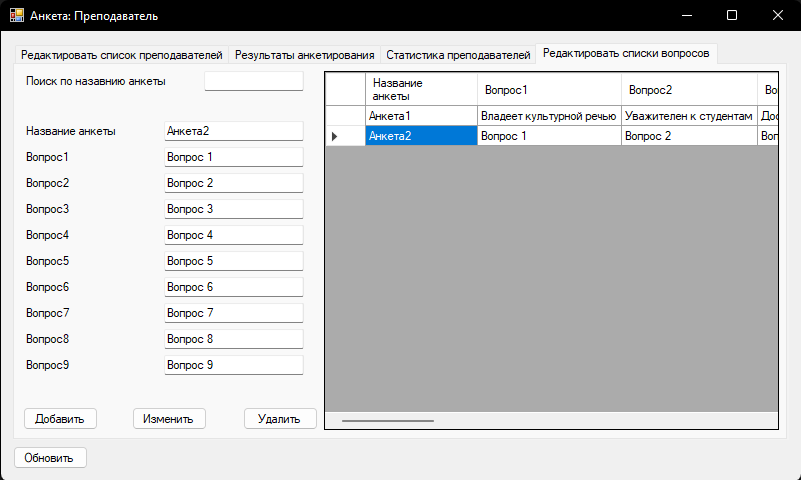


Рисунок 2.32 – Результат добавления данных в таблицу вопросов

Для изменения анкет в базе данных пользователю необходимо выбрать запись в таблице, значения которой скопируются в TextBox, после чего их можно будет редактировать, затем нажать на кнопку «Изменить», и выбранная запись будет изменена в базе данных. Результат изображён на рисунках 2.33.

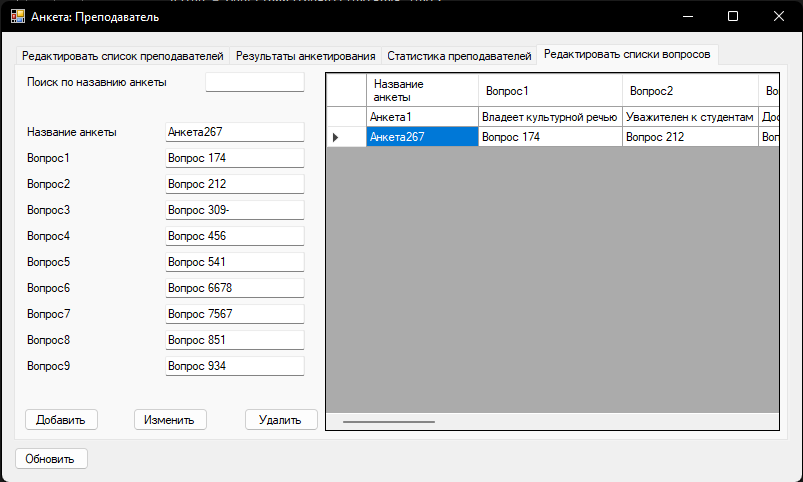
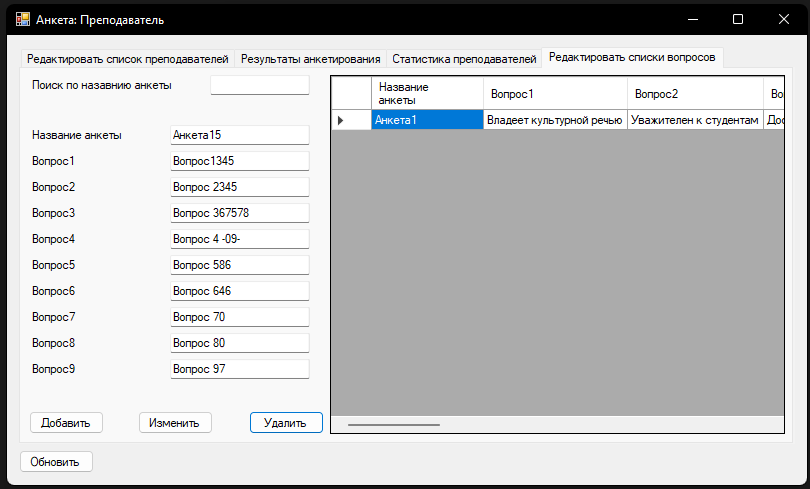


Рисунок 2.33 – Результат изменения данных в таблице вопросов

Для удаления анкеты нужно выбрать запись из таблицы и нажать на кнопку «удалить». Результат изображён на рисунке 2.34.

 Рисунок 2.34 – Результат удаления данных в таблице вопросов

# Заключение

В результате выполнения дипломной работы были выбраны и исследованы средства разработки:

* Платформа .NET Framework;
* Язык программирования C#;
* СУБД MySQL.

Создан интерфейс клиента, который состоит из окна входа, окна для студентов и окна для преподавателей, созданы концептуальная, логическая и физическая схемы базы данных, разобраны алгоритмы соединения, создания, изменения и чтения таблиц базы данных со стороны клиентской части приложения, а так же создан клиент, который может управлять записями в базе данных.

В созданном клиенте были разобраны все перечисленные ниже методы:

1. Открытие соединения;
2. Закрытие соединения;
3. Вывод значений из базы данных;
4. Добавление новых записей;
5. Изменение существующих записей;
6. Удаление существующих записей;
7. Расчёт статистики значений из таблицы.

# Список использованных источников

1. Гурвиц Г. Разработка реального приложения в среде клиент-сервер –ДВГУПС, 2005. 206 с. (дата обращения 01.05.2022)
2. Стиллмен, Э. Изучаем C#. / Э. Стиллмен, Дж. Грин. – 3-е изд. – СПб.: Питер, 2014. 816 с. (дата обращения 01.05.2022)
3. Подбельский, В.В. Язык C#. Базовый курс. / В.В. Подбельский – 2-е изд. – Москва: Инфра-М, 2013. 426 с. (дата обращения 01.05.2022)
4. Тепляков, С.В. Паттерны проектирования на платформе .NET. / С.В. Тепляков – СПб.: Питер, 2015. 320 с. (дата обращения 01.05.2022)
5. Ржеуцкая, С.Ю. Базы данных. Язык SQL: учеб.пособие / С.Ю. Ржеуцкая – Вологда: ВоГТУ, 2010. 159с. (дата обращения 01.05.2022)
6. .NET Framework – Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/.NET_Framework> (дата обращения 01.05.2022)
7. Общие сведения о платформе .NET [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview> (дата обращения 01.05.2022)
8. ADO.NET Entity Framework – Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/ADO.NET_Entity_Framework> (дата обращения 01.05.2022)
9. Общие сведения об Entity Framework [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/ef/overview> (дата обращения 01.05.2022)
10. Краткий обзор языка C# [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/> (дата обращения 17.06.2022)
11. MySQL – Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/MySQL> (дата обращения 17.06.2022)
12. СУБД MySQL – Метод Лаб [Электронный ресурс]. URL: <https://www.methodlab.ru/technology/mysql.shtml> (дата обращения 16.06.2022)
13. Студия Михаила Кечинова Что такое базы данных. Этапы создания баз данных. [Электронный ресурс]. URL: <https://mkechinov.ru/article.database.html> (дата обращения 17.06.2022)
14. Проектирование баз данных – Википедия [Электронный ресурс]. URL: <https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B0%D0%B7_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85> (дата обращения 18.06.2022)
15. Connect C# to MySQL – CodeProject [Электронный ресурс]. URL: <https://www.codeproject.com/Articles/43438/Connect-C-to-MySQL> (дата обращения 19.06.2022)
16. Документация по .NET | Microsoft Docs [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-RU/dotnet/> (дата обращения 21.06.2022)

# Приложение А

(Справочное)

Код алгоритма для нахождения среднего значения

public static void CountStatistics()

{

//Открытие соединения

MySqlConnection connection = MySql.OpenConnection();

//Создание текста команды

string commandText = "SELECT `Teacher\_Id`," +

"`Department\_Id`,`Answer1`,`Answer2`,`Answer3`," +

"`Answer4`,`Answer5`,`Answer6`,`Answer7`,`Answer8`," +

"`Answer9` FROM answers, teacher WHERE answers.Teacher\_Id" +

" = teacher.Id";

//Создание команды

MySqlCommand command = new MySqlCommand(commandText, connection);

//Создание адаптера

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

DataTable table = new DataTable();

//Заполнение таблицы адаптером

adapter.Fill(table);

//Создание числового массива

double[,] tableArray = new double[table.Rows.Count, 4];

//Упрощение переменных x-вертикаль y-горизонталь

int x = table.Rows.Count;

int y = table.Columns.Count;

//Заполнение числового массива значениями из таблицы

for (int i = 0; i < x; i++)

{

tableArray[i, 3] = 0;

for (int j = 0; j < y; j++)

{

if (j >= 2)

{

//Запись суммы ответов на вопросы

tableArray[i, 2] += double.Parse(table.Rows[i][j].ToString());

}

else

{

//Запись уникальных идентификаторов из таблицы в массив

tableArray[i, j] = double.Parse(table.Rows[i][j].ToString());

}

}

}

for (int i = 0; i < x; i++)

{

//Разделить сумму на количество элементов

tableArray[i, 2] = Math.Round((tableArray[i, 2] / 9), 2);

}

//Создание временного массива

double[,] tempArray;

//найти количество повторений преподавателей и вычислить их суммы

for (int i = 0; i < x-1; i++)

{

for (int j = i + 1; j < x; j++)

{

if (tableArray[i, 0] == tableArray[j, 0])

{

tableArray[i, 3] += 1;

tableArray[i, 2] = tableArray[j, 2] + tableArray[i, 2];

x--;

tempArray = new double[x, 4];

for (int k = 0; k < j; k++)

{

for (int c = 0; c < 4; c++)

{

tempArray[k, c] = tableArray[k, c];

}

}

for (int k = j; k < x; k++)

{

for (int c = 0; c < 4; c++)

{

tempArray[k, c] = tableArray[k+1, c];

}

}

tableArray = new double[x, 4];

for (int k = 0; k < x; k++)

{

for (int c = 0; c < 4; c++)

{

tableArray[k, c] = tempArray[k,c];

}

}

j--;

}

}

}

for (int i = 0; i < x; i++)

{

if (tableArray[i, 3] != 0)

{

//Разделить сумму на количество повторений записей + 1

tableArray[i, 2] = Math.Round(tableArray[i, 2] / (tableArray[i, 3] + 1),2);

}

}

for (int i = 0; i < x; i++)

{

//Создать команду поиска записи по Teacher\_Id

command = new MySqlCommand("SELECT `Teacher\_Id` FROM `statistics` WHERE `Teacher\_Id` LIKE @TeacherId", connection);

//Задать значение заглушке

command.Parameters.Add("@TeacherId", MySqlDbType.Int32).Value = tableArray[i, 0];

//Создать DataReader

MySqlDataReader reader = command.ExecuteReader();

int teacherId = 0;

while (reader.Read())

{

teacherId = reader.GetInt32(0);

}

reader.Close();

Console.WriteLine(teacherId);

//Проверить существование записи, если записи нет в таблице, то добавить её

if (teacherId==0)

{

//Создать команду добавления записи статистики

MySqlCommand addCommand = new MySqlCommand($"INSERT INTO `statistics` (`Teacher\_Id`,`AverageValue`) VALUES (@teacher\_id,@averageValue)", connection);

addCommand.Parameters.Add("@teacher\_id", MySqlDbType.Double).Value = tableArray[i, 0];

addCommand.Parameters.Add("@averageValue", MySqlDbType.Double).Value = tableArray[i, 2];

//Выполнить команду

addCommand.ExecuteNonQuery();

}

else

{

//Создать команду обновления статистики

command = new MySqlCommand($"UPDATE `statistics` SET `AverageValue` = @averageValue WHERE `Teacher\_Id` LIKE '%" + teacherId + "%'", connection);

command.Parameters.Add("@averageValue", MySqlDbType.Double).Value = tableArray[i, 2];

//Выполнить команду

command.ExecuteNonQuery();

}

}

MySql.CloseConnection(connection);

}