Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ИНСТИТУТ ЭЛЕКТРОННЫХ И ИНФОРМАЦИОННЫХ СИСТЕМ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

ПРИЛОЖЕНИЕ ДЛЯ КОНТРОЛЯ ЗАДАЧ

Отчет по работе по учебной дисциплине «Практика учебная: Практика ознакомительная»

по специальности 09.03.01 Информатика и вычислительная техника  
ИЭИС.КП 3091.004.42

|  |  |
| --- | --- |
|  | Руководитель:  Ст. препод. каф. ИТС  / Г. А. Архипова  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  | Студент группы 3091  / Р. А. Михайлов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Содержание

[Введение 3](#_Toc167705318)

[1. Постановка задачи 5](#_Toc167705319)

[1.1 Обоснования необходимости разработки 5](#_Toc167705320)

[1.2 Технико-математическое описание задачи 5](#_Toc167705321)

[1.3 Анализ предметной области 5](#_Toc167705322)

[1.4 Требования к программе 6](#_Toc167705323)

[1.5 Обоснование проектный решений 6](#_Toc167705324)

[1.6 Обзор и анализ существующих программный систем 7](#_Toc167705325)

[1.7 Вывод предпроектного исследования 8](#_Toc167705326)

[2. Теоретическая часть 9](#_Toc167705327)

[2.1 История научных исследований по выбранной теме. 9](#_Toc167705328)

[2.2 Определения и анализ ключевых терминов. 12](#_Toc167705329)

[2.3 Актуальные взгляды на выбранную тему. 15](#_Toc167705330)

[3. Практическая часть 17](#_Toc167705331)

[3.1 Анализ задачи 17](#_Toc167705332)

[3.2 Описание логической структуры 18](#_Toc167705333)

[3.3 Разработка программы для работы – входные и выходные данные 21](#_Toc167705334)

[3.4 Сравнительная характеристика работы 23](#_Toc167705335)

[Заключение 25](#_Toc167705336)

[Список литературы 26](#_Toc167705337)

[Приложение A 27](#_Toc167705338)

[Приложение Б (дополнительное) 29](#_Toc167705339)

[Приложение В (обязательное) 31](#_Toc167705340)

[Приложение Г (обязательное) 36](#_Toc167705341)

# Введение

На сегодняшний день, в условиях постоянного объемного расширения данных, эффективное управление базами данных является приоритетной задачей для многих организаций. Автоматизация хранения и обработки информации в базах данных критически важна для обеспечения оперативного доступа к данным и оптимизации бизнес-процессов. Разработка системы управления базами данных на основе структур с функционалом ввода и вывода данных в файл является актуальной и позволяет решить текущие проблемы работы с информацией.

Цель проекта заключается в разработке приложения для контроля задач с базой данных для хранения информации о проектах или задачах (название, описание, статус, ответственный и срок сдачи). Система предназначена для оптимизации процесса управления информацией, обеспечивая удобный и эффективный доступ к данным.

Целью данного проекта является создание приложения для контроля задач с базой данных на основе Microsoft SQL Server, способной обеспечить надежное хранение и обработку данных, а также возможность их сохранения в файл, с графическим интерфейсом под операционную систему Microsoft Windows на языке программирования C++/CLI.

Пользовательской аудиторией данного приложения может быть любой человек, владеющий навыками работы с компьютером, в частности школьники и студенты, у которых есть постоянная необходимость контролировать выполнение поставленных задач и проектов. Кроме того, эта база данных представляет потенциал для дальнейшего развития и применения в различных областях, где требуется эффективное управление информацией с использованием файловой системы.

Цель: разработка и реализация программы для работы с базой данных.

Объект: компьютерное приложение для контроля выполнения задач и проектов.

Предмет: база данных, соответствующая требованиям функциональности, включая хранение и управление информацией об учениках, заданиях и статистике прохождения тестирований.

Задачи:

1. Анализ аналогов, существующих БД;
2. Проектирование архитектуры системы с учетом требований к функциональности;
3. Написание кода для работы с базой данных и реализация операций ввода/вывода данных;
4. Тестирование системы на корректность работы и эффективность обработки информации.
   * + 1. Постановка задачи
   1. Обоснования необходимости разработки

В соответствии с заданием, требуется разработать приложение для хранения информации о проектах или задачах (название, описание, статус, ответственный и срок сдачи). Система предназначена для оптимизации процесса управления информацией, обеспечивая удобный и эффективный доступ к данным.

При создании будет использоваться база данных на основе Microsoft SQL Server, способной обеспечить надежное хранение и обработку данных, а также возможность их сохранения в файл.

* 1. Технико-математическое описание задачи

При разработке приложения будет использоваться язык программирования C++, для написания графического интерфейса будет использоваться язык программирования C++/CLI и платформа Microsoft Windows Forms. Для работы с базой данных будет использован языка программирования SQL. Пространства имен: «System::SqlClient», «System::IO», «System::VisualBasic», «System::ServiceProcess».

* 1. Анализ предметной области

Пользовательской аудиторией данного приложения может быть любой человек, владеющий навыками работы с компьютером, в частности школьники и студенты, у которых есть постоянная необходимость контролировать выполнение поставленных задач и проектов. Кроме того, эта база данных представляет потенциал для дальнейшего развития и применения в различных областях, где требуется эффективное управление информацией с использованием файловой системы.

* 1. Требования к программе

Требования к составу и параметрам технических средств, к операционной системе, возможно какие фреймворки необходимы, будет ли установщик ПО для desktop приложений, если это сайт, то требования к хостингу, пропускная способность и др.

* 1. Обоснование проектный решений
     1. Обоснование выбора языков программирования

Основанием для выбора языком программирования C++ послужило то, что он является основным изучаемым языком программирования. Его использование при разработке этого проекта – развитие практических навыков использования его в разных проектах.

Основание для выбора языка программирования SQL послужило то, что он является нативным для всех баз данных, включая Microsoft SQL Server.

Основание для выбора языка программирования C++/CLI послужило то, что он легко интегрируется с C++ и подходит для написания графического интерфейса на платформе Microsoft Windows Forms.

* + 1. Инструментальные средства

Для работы с базой данных выбрана система управления базами данных Microsoft SQL Server. Для открытия служебных файлов с расширение «.dbrom» и файлов вывода данных использовано стандартное приложение операционной системы Microsoft Windows Блокнот. Для создания иконки приложения использован графический редактор Adobe Photoshop и онлайн сервис векторный редактор.

* + 1. Обоснование выбора среды программирования

Для разработки приложения выбрана интегрированная среда разработки (IDE) Microsoft Visual Studio. Обоснованием для выбора этой среды разработки выступает удобство программирования в ней C++. У данной среды разработки существует множество доступным расширений, которые пригодились при разработке этого проекта, а именно возможность реализация графического интерфейса, простое подключение к базе данных и быстрое создание установщика полученного приложения.

* + 1. Информационное обеспечение

При работе над проектом были использованы такие программы средства, как:

– Microsoft Word (написание отчета о проделанной работе);

– Microsoft Power Point (создание презентации);

– darw.io (создание схемы).

* 1. Обзор и анализ существующих программный систем

Одним из аналогов приложения является Microsoft Projects. Достоинством этого аналога является его функциональная обширность, в нем есть достаточно большое количество функций, позволяющий проводить манипуляции над записями (задачами и проектами). Недостатком является сложный в понимании интерфейс, перегруженный большим количеством кнопок и выпадающих списков, препятствующие интуитивному пониманию программы.

Следующим аналогом приложения выступает Wrike. Достоинство этого аналога наследует у Microsoft Projects, такое же большое количество функций, интерфейс у этого сервиса приятнее, чем у предыдущего аналога. Недостатком является хранение данных в облачном хранилище.

Еще одним аналогом приложения можно назвать платформу Comindware. Достоинством этой платформы можно назвать такое интересное решение, как использование не стандартного варианта представления данных в таблице, а использование блок-схем. На платформе есть все необходимы функции для работы с задачами и проектами.

Недостаток каждого из аналогов, за исключением Microsoft Projects, заключается в том, что для их функционирования необходимо Интернет-подключение, так как они хранят информацию о проектах и задачах на удаленном сервере, в отличии от реализованного в процессе выполнения этого проекта приложения, где информация хранится в локальных файлах.

* 1. Вывод предпроектного исследования

Цель разработки прикладного программного обеспечения заключается в создании удобного приложения для контроля проектов и задач, обеспечивающее визуальное представление запланированной работы и возможность рационально управлять своим временем (занятостью).

1. Теоретическая часть
   1. История научных исследований по выбранной теме.
      1. Модель Эдгара Фрэнка Кодда

Первые компьютерные базы данных появились в конце 1960-х годов. В то время это была важная область исследований. Многие ученые в области информатики были сосредоточены на улучшении работы баз данных. Одним из них был Эдгар Фрэнк (Тед) Кодд, английский специалист по вычислительной технике, работавший в IBM.

В то время, когда Кодд писал свою статью, доминировали иерархические и сетевые базы данных. Они также были довольно негибкими. Чтобы извлечь данные из базы данных, вам, по сути, нужно было написать компьютерную программу: данные были недоступны для непрограммистов. Любое изменение в модели требовало изменений в схемах доступа к данным – другими словами, программы доступа к данным в основном приходилось переписывать.

В своей статье Кодд предложил совершенно новую идею: моделирование данных с помощью математического понятия отношений (сегодня мы называем их таблицами). Реляционная модель данных Кодда обеспечивает большую гибкость, чем иерархическая и сетевая модели данных. Новые отношения могут быть добавлены без изменения существующих отношений. Благодаря его идеям работа с базами данных стала намного проще.

Модель Кодда не сразу стала успешной. IBM не стремилась внедрять его предложения. В то время у них была IMS, очень успешная иерархическая база данных. Они не хотели снижать свои доходы от IMS, создавая конкурирующий продукт. (IMS разрабатывается и по сей день, что показывает, насколько успешной она была.) Только в 1973 году IBM запустила System R - исследовательский проект по изучению идей Кодда в отношении реляционной модели данных. Кодд не работал в тесном контакте с командой System R; трудно понять, почему его отстранили от проекта, основанного на его собственной работе. Два человека, участвовавшие в разработке System R, Дон Чемберлин и Рэй Бойс, отвечали за создание языка запросов.

В своей оригинальной статье Кодд предложил набор операций, которые можно было бы использовать для извлечения данных из отношений. Мы можем рассматривать эти операции как первый язык запросов для реляционных баз данных. Конечно, синтаксис был совершенно иным, чем у SQL, который мы знаем сегодня; Кодд использовал математическое описание для этого языка. Большинство операций, предложенных Коддом, могут быть выполнены в современном SQL, просто с другой форме.

В то время Дон Чемберлин работал над иерархическими базами данных и изучил язык запросов к этим базам данных.

Фактически, Кодд предложил два разных языка для реляционной модели: реляционную алгебру (основа для этого языка была заложена в его оригинальной статье 1970 года) и реляционное исчисление (также известное как язык Alpha). Оба этих языка использовали математическую формулу с кванторами и различными математическими операторами.

* + 1. QUEL, SEQUEL, SQUARE

Реляционное исчисление/Alpha стало основой для QUEL, языка запросов для Ingres (Interactive Graphics Retrieval System – Интерактивной Графической Поисковой Системы), ранней реляционной базы данных, разработанной Майклом Стоунбрейкером из Калифорнийского университета в Беркли. Ingres превратился во многие коммерческие приложения для работы с базами данных, такие как PostgreSQL.

Еще до начала проекта System R Чемберлин и Бойс разработали язык, который они назвали SQUARE (Specifying Queries as Relational Expressions – Определяющий Запросы Как Реляционные Выражения). Они оценили силу идей Кодда, которые позволили им использовать несколько строк для выражения сложных запросов, которые занимали бы страницы в иерархической базе данных. Они были убеждены, что их язык проще и доступнее для обычных пользователей, чем реляционная алгебра и реляционное исчисление Кодда. (Сравнение записи запроса в Приложении А.1).

SQUARE стал основой для нового языка запросов System R. в SQUARE использовалось много индексов и некоторые математические обозначения. Его было сложно печатать на клавиатуре. Чемберлин и Бойс решили адаптировать его таким образом, чтобы он напоминал структуру английского предложения и был проще для ввода.

При разработке SEQUEL для Чемберлина и Бойса были важны две вещи. Во-первых, они хотели, чтобы программа была доступна обычным пользователям, не имеющим математического образования или опыта программирования. Сотрудники System R даже набрали группу студентов, чтобы они изучили SEQUEL и посмотрели, насколько прост синтаксис. Кроме того, они хотели, чтобы язык содержал элементы модификации и определения данных, что в то время было чем-то совершенно новым.

* + 1. DML, DDL, SQL

Наконец, Чемберлин и Бойс написали две статьи о SEQUEL: одну о DML (Data Manipulation Language, e.g. SELECT, INSERT, and UPDATE – языке манипулирования данными, например, инструкции SELECT, INSERT и UPDATE) и одну о DDL (Data Definition Language – языке определения данных, который используется для создания и изменения структуры базы данных).

Так и родился SEQUEL. Позже SEQUEL был переименован в SQL из-за проблемы с торговой маркой.

С годами SQL стал отраслевым стандартом. На данный момент достаточно сказать, что SQL стал базовым языком для работы с базами данных. Он признан всеми важными организациями, а такие гиганты рынка, как Google и Facebook, ежедневно используют его для многих процессов.

* 1. Определения и анализ ключевых терминов.
     1. Описание программных компонентов

С++/CLI — это набор расширений языка C++. С применением расширений C++/CLI можно определять управляемые и неуправляемые классы и функции в пределах одного файла исходного кода, использовать управляемые и низкоуровневые типы C и C++, как в обычном программном коде на C++, то есть простым подключением заголовочного файла и связыванием с библиотекой. Данных язык совместим с Microsoft .NET Framework, а следовательно, и с Windows Forms.

Microsoft SQL Server — это реляционная система управления базами данных (RDBMS). Приложения и средства подключаются к экземпляру или базе данных SQL Server и взаимодействуют с помощью Transact-SQL (T-SQL).

Windows Forms — это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений.

* + 1. Понятие БД и СУБД

База данных (БД) — это набор информации, которая хранится упорядоченно в электронном виде.

Системы управления базами данных (СУБД) — это инструменты, с помощью которых пользователь обращается к данным, изменяет их или создаёт. СУБД функционируют как менеджеры по работе с базами данных, которые «говорят» на их языке программирования.

* + 1. Свойства БД

Удобное использование баз данных основано на их свойствах:

1. Быстродействие

Современные БД проектируются по принципу «получить данные прямо сейчас», чтобы пользователь не ждал отклик на запрос.

1. Простота получения и обновления данных

Какой бы высокой ни была скорость, это бессмысленно, если нужно сделать много сложных операций, чтобы получить, обновить или добавить данные в базу.

1. Независимость структуры

Изменения в любом количестве и качестве информации не должны влиять на структуру базы данных. Также изменения не должны касаться программного обеспечения и средств хранения, например жёсткого диска.

1. Стандартизация

Аналогично свойству независимости структуры: при обновлении программного обеспечения или СУБД база данных не должна менять свою структуру или свойства.

1. Безопасность данных

Для каждой категории пользователей делают список ограничений и доступов, согласно которым можно взаимодействовать с информацией из БД.

1. Интегрированность

Данные должны быть логически связаны. И эти связи должны прослеживаться по структуре таблиц.

* + 1. Используемы операторы SQL

SQL содержит большое количество различным операторов, для выполнения этой работы, я использовал следующие операторы:

DDL (операторы определения данных):

– CREATE TABLE (оператор создания таблицы), создание новой таблицы в базы данных;

– DROP TABLE (оператор удаления таблицы), удаляет таблицу из базы данных.

DML (операторы манипуляции данными):

– DELETE (оператор удаление строки), удаляет одну или несколько строк, соответствующих условиям фильтрации, из базовой таблицы. Применение оператора согласуется с принципами поддержки целостности, поэтому этот оператор не всегда может быть выполнен корректно, даже если синтаксически он записан правильно;

– INSERT (оператор добавления строки), вставляет одну строку в базовую таблицу. Допустимы модификации оператора, при которых сразу несколько строк могут быть перенесены из одной таблицы или запроса в базовую таблицу;

– UPDATE (оператор обновления строки), обновляет значения одного или нескольких столбцов в одной или нескольких строках, соответствующих условиям фильтрации;

– SELECT (оператор выборки), оператор, заменяющий все операторы реляционной алгебры, и позволяющий сформировать результирующее отношение, соответствующее запросу.

* 1. Актуальные взгляды на выбранную тему.
     1. Технологический прогресс

В современном мире технологический прогресс является неотъемлемой частью бизнес-процессов.

Автоматизация и централизация управления задачами и проектами через специализированные приложения становятся все более востребованными.

* + 1. Оптимизация деятельности через приложения с графическим интерфейсом:

Приложения на базе Windows Forms обеспечивают удобный и интуитивно понятный пользовательский интерфейс, что способствует эффективной работе с данными и управлению задачами.

Эффективное использование языка C++/CLI позволяет создавать быстрые и производительные приложения, что важно для эффективной работы с данными в реальном времени.

* + 1. Значение баз данных для управления задачами и проектами:

Базы данных, такие как Microsoft SQL Server, играют ключевую роль в хранении, организации и управлении данными о задачах и проектах.

Использование SQL для работы с базой данных обеспечивает удобные возможности поиска, фильтрации и анализа информации, что необходимо для эффективного управления задачами и проектами.

* + 1. Интеграция современных технологий:

Интеграция создаваемого приложения с другими современными технологиями, такими как облачные сервисы, мобильные приложения и аналитические инструменты, способствует повышению его ценности и конкурентоспособности.

Тема разработки приложения для ОС Windows с использованием Windows Forms, языка C++/CLI, SQL и интеграцией с Microsoft SQL Server, направлена на современные требования к эффективному управлению задачами и проектами. Комбинация технологий позволит создать удобное инструментальное средство, способствующее оптимизации деятельности и повышению производительности.

1. Практическая часть
   1. Анализ задачи

Задачей проекта является создание приложения для контроля и управления задачами и проектами. При разработке решения задачи были определены следующие методы:

– Метод хранения информации о задачах и проектах – использование базы данных. Выбор такого метода обусловлен удобством использования баз данных, а именно структурированностью хранимой информации и способа получения данных из БД, посредством языка структурированных запросов (SQL). При создании приложения в качестве СУБД была выбрана Microsoft SQL Server, так как при работе в среде разработки Microsoft Visual Studio удобнее всего использовать именно её;

– Метод написания логики приложения – написание кода на языке программирования C++/CLI. Выбор данного метода обусловлен тем, что этот язык программирования является основным при обучении, его использование в данном проекте – хорошая практика. Кроме того, у меня есть опыт создания программ на этом языке программирования, то есть при его использовании будет обеспечена максимальная корректность выполнения программы;

– Метод создания графического интерфейса – разработка при помощи платформы Microsoft Windows Forms. Выбор этого метода обусловлен тем, что при работе с ней в среде разработки Microsoft Visual Studio обеспечена высокая степень интеграции, удобный инструмент при разработке приложений с графическим интерфейсом.

* 1. Описание логической структуры
     1. Алгоритм программы

Описание всех алгоритмов программы. В целом о работе программы и подробно рассказать про какие-то сложные или интересные моменты (какая-то сложная логика, функция).

Программа написана в стиле объектно-ориентированного программирования. Часть исходного кода сгенерировано Microsoft Visual Studio, код графического интерфейса. В проект входит несколько заголовочных файлов с реализациями классов (окон приложения):

– MainWindow.h (Основное окно, вывод таблицы базы данных), через это окно ведется работа с базой данных. В окне присутствует две зоны: меню и поле вывода таблицы базы данных. В меню определены кнопки управления: «Подключение», «Редактирование», «Запрос», «Вывод в файл». Листинг файла MainWindow.h находится в Приложении Г;

– ConnectionWindow.h (Окно подключения к базе данных), через это окно происходит процесс подключения к базе данных, ввод необходимых параметров. При успешном подключении параметры подключения сохраняются в файл в папку с исполняемым файлом в пользовательском расширении «.dbrom»;

– NewRecord.h (Окно создания новой записи), через это окно происходит процесс создания новой записи, в заголовочном файле реализована обработка ошибок, некорректного ввода данных;

– DeleteRecord.h (Окно удаления записи), через это окно происходит процесс удаления записи, для удаления записи в выпадающем списке необходимо выбрать ID записи для удаления, приложение получает ID автоматически, при помощи SQL запроса;

– EditRecord.h (Окно редактирования записи), через это окно происходить процесс редактирования записи, редактирование реализовано точечно, то есть в один момент можно редактировать только одну ячейку. Для редактирования необходимо выбрать ID записи и редактируемую колонку (в отдельном поле выводится текущее значение ячейки), проложение получает данных автоматически, при помощи SQL запроса.

Открытие дополнительных окон осуществляется при помощи обработчика событий, при нажатии кнопки срабатывает обработчик и запускается определенная функция (процедура). Каждое окно, в контексте исходного кода представляет собой экземпляр класса, наследуемого от класса «System::Windows::Forms::Form».

В каждом классе реализованы внутренние методы, выполняющие конкретные действия. Все методы, за исключением методов по получению значений полей классов с модификатором доступа «private», имеют модификатор доступа «private».

Методы, имеющие модификатор доступа «public» выполняют только функции передачи значения из одного окна (класса) в другое. Структура приложения реализована в такой форме, когда основное окно (MainWindow) может вызвать любое другое, а все остальные окна могут только проводить какие-либо манипуляции над таблицей базы данных.

Взаимодействие приложения с базой данных реализовано при помощи пространства имён «System::Data::SqlClient».

Для подключения к базе данных создается экземпляр класса «SqlConnection», в качестве аргумента конструктора класса выступает строка подключения в формате «Data Source = название сервера; Initial Catalog = название базы данных; Integrated Security = True; Trust Server Certificate = True». Такой экземпляр класса определен в программе как глобальная переменная.

Для работы с базой данных, создание запросов, создается экземпляр класса «SqlCommand», где в качестве аргумента конструктора выступает заранее определенный строковый запрос типа данных System::String.

Для вывода таблицы базы данных на экран используется элемент графического интерфейса dataGridView. Вывод в dataGridView реализован посредством запроса к базе данных, результат которого представлен в виде экземпляра класса «SqlDataReader». Полученный данных выводятся на экран при помощи процедуры read\_single\_row(), выводящей данные построчно.

Для вывода данных в текстовый файл, а так же для сохранения параметров последнего успешного подключения, было использовано пространство имен «System::IO», частью которого является класс «StreamWirter», используемый для записи в файл. В качестве аргументоа конструктора класса выступает строковая переменная тип System::String, а именно абсолютный путь для сохранения файла, полученный при помощи элемента интерфейса «SaveFileDialog».

В исходном коде класса ConnectionWindow реализован запуск и остановка службы локального SQL-сервера при помощи пространства имен «System::ServiceProcess». Для запуска и остановки службы, имя которой вводятся при запуске приложения, используется экземпляр класса «ServiceController», к качестве аргумента конструктора класса выступает строковая переменная, хранящая имя службы локального SQL-сервера.

При закрытии основного окна (MainWindow) запускается деконструктор класса MainWindow, который предлагает остановить службу локального SQL-сервера. Также деконструктор освобождает память от всех элементов графического интерфейса, в частности классов дополнительный окон, запуская их деконструкторы.

* + 1. Составные части программы и связи между ними

Схематическое изображение взаимодействия модулей программы между собой представлено в Приложении А.2.

## Разработка программы для работы – входные и выходные данные

На вход подается информация о задаче (проекте), запись состоит из таких пунктов, как: ID записи, название задачи, описание задачи, статус (этап выполнения), ФИО ответственного и срок сдачи проекта/задачи. На выход подается таблица базы данных с введенной ранее информацией о задаче (проекте), данные представляются в виде таблицы.

### 3.3.3 Тестирование

Выполнено 5 тест-кейсов (таблицы с результатами в приложении Г) и 3 Unit-теста, встроенные в проект.

### 3.3.4 Эксплуатационные документы – руководство пользователя

### 3.3.4.1 Назначение программы

Пользовательской аудиторией данного приложения может быть любой человек, владеющий навыками работы с компьютером, в частности школьники и студенты, у которых есть постоянная необходимость контролировать выполнение поставленных задач и проектов. Кроме того, это приложение представляет потенциал для дальнейшего развития и применения в различных областях, где требуется эффективное управление информацией с использованием файловой системы.

### 3.3.4.2 Условия выполнения

Компьютер, на котором установлена 64–битная операционная система Microsoft Windows 10 или Microsoft Windows 11. Также на компьютере должна быть установлена программа для работы с базой данных, а именно Microsoft SQL Server, установщик СУБД находится в папке с исполняемым файлом.

### 3.3.4.3 Выполнение программы

Для начала работы с приложение необходимо запустить службу локального SQL-сервера, её можно включить, как вручную, так и через приложение, введя ее название. После запуска службы необходимо подключиться к таблице базы данных, для этого в соответствующие поля окна подключения, оно открывается при запуске, нужно ввести названия сервера и базы данных (при успешном подключении они сохраняются в файл в папку с исполняемым файлом). При наличии таблицы, соответствующей условиям подключения, а именно:

– У колонок должны быть следующие названия, сначала идет имя колонки, в скобках тип данных: ID (int), label (varchar(200)), description (varchar(200)), status (varchar(200)), admin (varchar(200)), deadline (date);

– У колонки ID должен быть включён параметр "Identity Specification";

– У всех колонок должен быть отключён параметр "Allow Nulls".

после нажатия на checkbox «Подключиться к существующей таблице», в выпадающем списке будут перечислены подходящие таблицы, которые можно выбрать, после чего, нажав кнопку «Подключиться», окно подключения будет закрыто, а данные из таблицы выведены на экран.

При отсутствии такой таблицы ее можно создать вручную через Microsoft SQL Management Studio, который необходимо установить отдельно, или через встроенную в приложение функцию. Для создания такой таблицы через приложение в поле «Создать новую таблицу» достаточно ввести только её имя и нажать кнопку «Подключиться».

После успешного подключения можно начать работу с таблицей базы данных:

– В разделе «Редактирование» доступны такие функции, как: добавление новой записи (при создании новой записи на ввод данных установлены ограничения: максимальная длина вводимого текста – 200 символов, минимальный строк сдачи 01.01.2024), удаление записи по ID, удаление всех записей из таблицы, редактирование записи по ID и удаление текущей таблицы (редактирование записи происходит точечно, а именно по ячейкам таблицы, то есть по пересечению ID записи с выбранной колонкой);

– В разделе «Запрос» доступны такие варианты запросов, как: вывод всей таблицы, вывод таблицы по статусу, вывод таблицы по сроку сдачи (в прямом и обратном порядке), вывод таблицы по ответственному (в прямом и обратном порядке) и ввод пользовательского запроса (требует доработки);

– В приложении доступна функция вывода данных из таблицы базы данных в текстовый файл, для этого необходимо нажать на «Вывод в файл» и выбрать место для сохранения.

Скриншоты окно приложения находятся в Приложении Б.

* 1. Сравнительная характеристика работы

Одним из аналогов приложения является Microsoft Projects. Достоинством этого аналога является его функциональная обширность, в нем есть достаточно большое количество функций, позволяющий проводить манипуляции над записями (задачами и проектами). Недостатком является сложный в понимании интерфейс, перегруженный большим количеством кнопок и выпадающих списков, препятствующие интуитивному пониманию программы.

Следующим аналогом приложения выступает Wrike. Достоинство этого аналога наследует у Microsoft Projects, такое же большое количество функций, интерфейс у этого сервиса приятнее, чем у предыдущего аналога. Недостатком является хранение данных в облачном хранилище.

Еще одним аналогом приложения можно назвать платформу Comindware. Достоинством этой платформы можно назвать такое интересное решение, как использование не стандартного варианта представления данных в таблице, а использование блок-схем. На платформе есть все необходимы функции для работы с задачами и проектами.

Недостаток каждого из аналогов, за исключением Microsoft Projects, заключается в том, что для их функционирования необходимо Интернет-подключение, так как они хранят информацию о проектах и задачах на удаленном сервере, в отличии от реализованного в процессе выполнения этого проекта приложения, где информация хранится в локальных файлах.

Заключение

При работе над данным проектом были получены следующие результаты:

– Изучены и отработаны необходимые навыки работы с базами данных (Microsoft SQL Server) при помощи языка структурированных запросов SQL;

– Получены знания по подключению и работе с базой данных посредством языка программирования C++/CLI на платформе Microsoft Windows Forms;

– Освоена работа с приложением, состоящим из нескольких функциональных окон (передача данных между окнами);

– Реализовано приложение – база данных для контроля задач.

# Список литературы

1. Изучаем SQL: генерация, выборка и обработка данных / Алан Болье; перевод с английского и редакция И. В. Красикова. - 3-е изд. - Москва: Диалектика; Санкт-Петербург: Диалектика, 2021. - 400 с.: табл.; 24 см.; ISBN 978-5-907365-54-4
2. The History of SQL – How It All Began: Learn SQL – URL: [https://learnsql.com/blog/history-of-sql/](https://learnsql.com/blog/history-of-sql/" \o "https://learnsql.com/blog/history-of-sql/)
3. Документация по Microsoft SQL : Microsoft Learn – URL: [https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16](https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16" \o "https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16)
4. Как работают базы данных в IT: разбор на примерах : Блог Яндекс Практикум – URL: [https://practicum.yandex.ru/blog/chto-takoe-bazy-dannyh/](https://practicum.yandex.ru/blog/chto-takoe-bazy-dannyh/" \o "https://practicum.yandex.ru/blog/chto-takoe-bazy-dannyh/)
5. Программирование .NET с использованием C++/CLI : Microsoft Learn – URL: [https://learn.microsoft.com/ru-ru/cpp/dotnet/dotnet-programming-with-cpp-cli-visual-cpp?view=msvc-170](https://learn.microsoft.com/ru-ru/cpp/dotnet/dotnet-programming-with-cpp-cli-visual-cpp?view=msvc-170" \o "https://learn.microsoft.com/ru-ru/cpp/dotnet/dotnet-programming-with-cpp-cli-visual-cpp?view=msvc-170)
6. Руководство по классическим приложениям (Windows Forms .NET) : Microsoft Learn – URL: [https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-8.0](https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-8.0" \o "https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-8.0)
7. Справочник по Transact-SQL (ядро СУБД) : Microsoft Learn – URL: [https://learn.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver16](https://learn.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver16" \o "https://learn.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver16)

# Приложение A

(справочное)

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

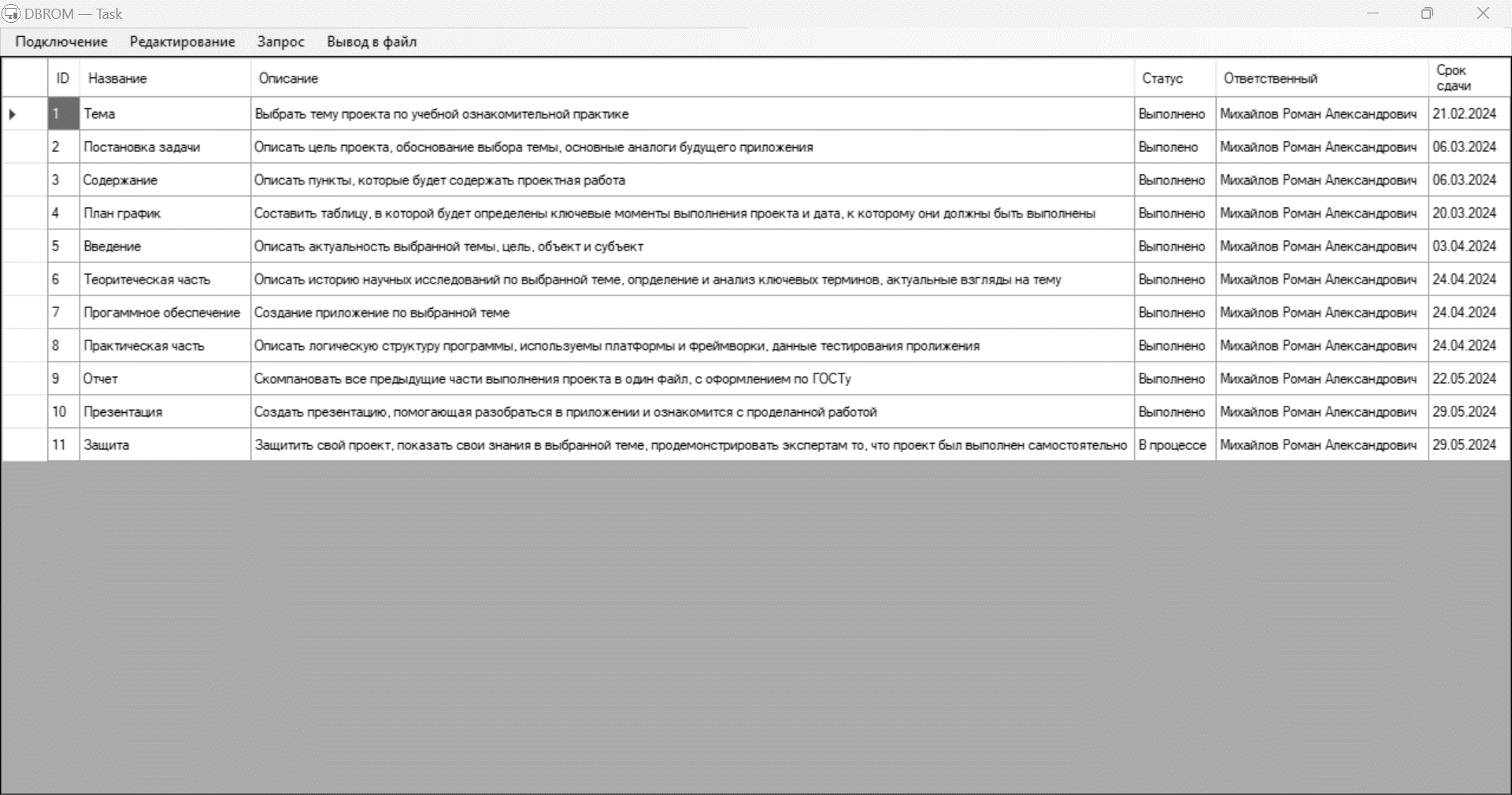
Автоматически созданное описание

Рисунок А.1 – сравнение записи запроса («Найдите имена сотрудников, которые зарабатывают больше, чем их руководители»)

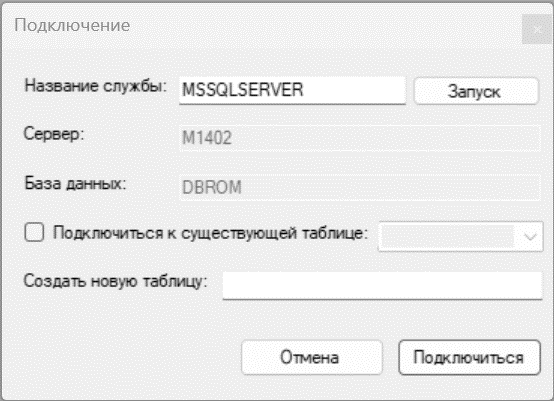
Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Автоматически созданное описаниеРисунок А.2 – схематическое изображение взаимодействия модулей программы между собой

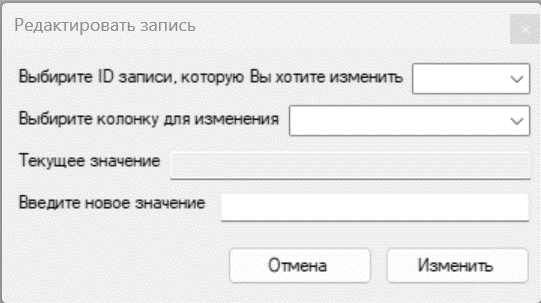
# Приложение Б (дополнительное)



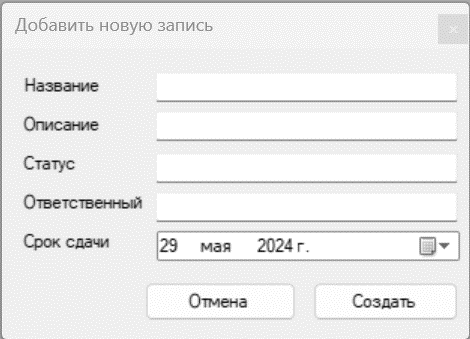
Приложение Б.1 Рисунок 3 – Главное окно приложения с примером использования



Приложение Б.2 Рисунок 4 – Окно подключения



Приложение Б.3 Рисунок 5 – Окно редактирования записи



Приложение Б.4 Рисунок 6 – Окно создания новой записи

# Приложение В (обязательное)

Таблица В.1 – тест-кейс №1.

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T01 |
| Заголовок | | | Проверка корректности подключения к существующей таблице |
| Предусловие | | | Приложение не запущено |
| Ожидаемый результат | | | Вывод данных из таблицы на экран |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Запустить DBROM.exe | Открытие основного окна и окна подключения | Pass |
| 2 | Ввод названия SQL-службы, сервера и БД | Активизация поля «Подключиться к существующей таблицы») | Pass |
| 3 | Выбор существующей таблицы | Вывод названия выбранной таблицы | Pass |
| 4 | Нажать кнопку «Подключиться» | Закрытие окна подключения, вывод данных на экран | Pass |
| Результат | | | Pass |

Таблица В.2 – тест-кейс №2

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T02 |
| Заголовок | | | Проверка функции создания новой записи |
| Предусловие | | | Приложение запущено, подключение к таблице выполнено |
| Ожидаемый результат | | | Добавление в БД новой записи |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Редактирование» | Открытие меню раздела | Pass |
| 2 | Выбрать пункт «Новая запись» | Открытие окна создания новой записи | Pass |
| 3 | Ввести данные в соответствующие поля | Форма заполнена | Pass |
| 4 | Нажать кнопку «Создать» | Закрытие окна добавления новой записи и вывод обновленной БД на экран | Pass |
| Результат | | | Pass |

Таблица В.3 – тест-кейс №3

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T03 |
| Заголовок | | | Проверка функции редактирования записи |
| Предусловие | | | Приложение запущено, подключения к таблице выполнено |
| Ожидаемый результат | | | Вывод отредактированной информации |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Редактирование» | Открытие меню раздела | Pass |
| 2 | Выбрать пункт «Редактирование записи» | Открытие окна редактирования записи | Pass |
| 3 | Выбор ID записи для редактирования | Вывод ID записи | Pass |
| 3 | Выбор колонки для редактирования | Вывод названия колонки, вывод текущих данных | Pass |
| 4 | Ввод новых данных и нажатие кнопки «Изменить» | Закрытие окна, вывод отредактированной информации на экран | Pass |
| Результат | | | Pass |

Таблица В.4 – тест-кейс №4

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T04 |
| Заголовок | | | Проверка функции вывода данных по сроку сдачи |
| Предусловие | | | Программа запущена, подключение к таблице выполнено |
| Ожидаемый результат | | | Вывод данных по сроку сдачи |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Запрос» | Открытие меню раздела | Pass |
| 2 | Выбрать пункт «Вывод таблицы по сроку сдачи» | Открытие подменю | Pass |
| 3 | Выбрать пункт «В прямом порядке» | Вывод данных в соответствии со сроком сдачи | Pass |
| Результат | | | Pass |

Таблица В.5 – тест-кейс №5

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | | | T05 |
| Заголовок | | | Проверка контроля ввода данных |
| Предусловие | | | Приложение запущено, подключение к таблице выполнено |
| Ожидаемый результат | | | Вывод сообщения о некорректности данных |
| Шаг | Действие | Предполагаемый результат | Pass/Fail |
| 1 | Выбрать раздел «Редактирование» | Открытие меню раздела | Pass |
| 2 | Выбрать пункт «Новая запись» | Открытие окна создания новой записи | Pass |
| 3 | Ввод в поле «Срок сдачи» некорректное значение, например 30.02.2024 | Вывод сообщения об ошибке | Pass |
| Результат | | | Pass |

# Приложение Г (обязательное)

Листинг файла MainWindow.h

#pragma once

#include "ConnectionWindow.h"

#include "NewRecord.h"

#include "EditRecord.h"

#include "DeleteRecord.h"

#include "ConnectionWindow.h"

#include "CloseBar.h"

namespace DBROM {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace System::Data::SqlClient;

using namespace Microsoft::VisualBasic;

using namespace System::IO;

using namespace Microsoft::VisualBasic;

using namespace System::ServiceProcess;

/// <summary>

/// Сводка для Visual

/// </summary>

public ref class MainWindow : public System::Windows::Forms::Form

{

public:

MainWindow(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

SaveFileDialog->Filter = "Текстовый файл (\*.txt)|\*.txt"/\*|PDF - файл (\*.pdf)|\*.pdf"\*/;

this->Show();

connection\_to\_constr();

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MainWindow()

{

try

{

int q = 0;

CloseBar^ cb = gcnew CloseBar();

cb->ShowDialog(this);

ServiceController^ service = gcnew ServiceController(serviceName);

if (service->Status != ServiceControllerStatus::Stopped)

{

if (q == 0)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Остановить службу локального SQL-сервера?", "Уведомление", MessageBoxButtons::YesNo, MessageBoxIcon::Question);

q = 1;

if (result == System::Windows::Forms::DialogResult::Yes)

{

service->Stop();

service->WaitForStatus(ServiceControllerStatus::Stopped, TimeSpan::FromMinutes(1));

if (service->Status == ServiceControllerStatus::Stopped)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Служба локального SQL-сервера остановлена", "Уведомление", MessageBoxButtons::OK, MessageBoxIcon::Information);

if (result == System::Windows::Forms::DialogResult::Yes) {}

}

}

}

}

}

catch (System::Exception^) {}

if (components)

{

delete components;

}

}

protected:

private: SqlConnection^ connection; // Строка подключения

private: String^ data\_source = ""; // Название сервера

private: String^ initial\_catalog = ""; // Название базы данных

private: String^ t\_name = ""; // Имя таблицы

private: ConnectionWindow^ cw = gcnew ConnectionWindow(true); // Окно подключения

private: String^ prev\_query = "";

private: String^ serviceName = "";

private: System::Windows::Forms::MenuStrip^ menuStrip1;

private: System::Windows::Forms::ToolStripMenuItem^ редактированиеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ запросToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выводВФайлToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ новаяЗаписьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ удалитьЗаписьToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ удалитьВсеЗаписиToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выводТаблицыПоОтветственномуToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выводТаблицыПоСтатусуToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выводТаблицыПоСрокуСдачиToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ выводТабToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ вПрямомПорядкеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ вОбратномПорядкеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ аЯToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ яАToolStripMenuItem;

private: System::Windows::Forms::DataGridView^ dataGridView1;

private: System::Windows::Forms::ToolStripMenuItem^ редактированиеЗаписиToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ подключениеToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ информацияToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ подключениеВБДToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ удалитьТаблицуToolStripMenuItem;

private: System::Windows::Forms::ToolStripMenuItem^ пользовательскийSQLзапросToolStripMenuItem;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column2;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column3;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column6;

private: System::Windows::Forms::ToolStripMenuItem^ выводВФайлToolStripMenuItem1;

private: System::Windows::Forms::SaveFileDialog^ SaveFileDialog;

private: System::Windows::Forms::ToolStripMenuItem^ закрытьПрограммуToolStripMenuItem;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MainWindow::typeid));

this->menuStrip1 = (gcnew System::Windows::Forms::MenuStrip());

this->подключениеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->информацияToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->подключениеВБДToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->закрытьПрограммуToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->редактированиеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->новаяЗаписьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->удалитьЗаписьToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->удалитьВсеЗаписиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->редактированиеЗаписиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->удалитьТаблицуToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->запросToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выводТабToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выводТаблицыПоСтатусуToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выводТаблицыПоСрокуСдачиToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->вПрямомПорядкеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->вОбратномПорядкеToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выводТаблицыПоОтветственномуToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->аЯToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->яАToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->пользовательскийSQLзапросToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выводВФайлToolStripMenuItem1 = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->выводВФайлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->Column1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column2 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column3 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column4 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column5 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->SaveFileDialog = (gcnew System::Windows::Forms::SaveFileDialog());

this->menuStrip1->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

this->SuspendLayout();

//

// menuStrip1

//

this->menuStrip1->GripMargin = System::Windows::Forms::Padding(2, 2, 0, 2);

this->menuStrip1->ImageScalingSize = System::Drawing::Size(24, 24);

this->menuStrip1->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(4) {

this->подключениеToolStripMenuItem,

this->редактированиеToolStripMenuItem, this->запросToolStripMenuItem, this->выводВФайлToolStripMenuItem1

});

this->menuStrip1->Location = System::Drawing::Point(0, 0);

this->menuStrip1->Name = L"menuStrip1";

this->menuStrip1->Size = System::Drawing::Size(1898, 33);

this->menuStrip1->TabIndex = 19;

this->menuStrip1->Text = L"menuStrip1";

//

// подключениеToolStripMenuItem

//

this->подключениеToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(3) {

this->информацияToolStripMenuItem,

this->подключениеВБДToolStripMenuItem, this->закрытьПрограммуToolStripMenuItem

});

this->подключениеToolStripMenuItem->Name = L"подключениеToolStripMenuItem";

this->подключениеToolStripMenuItem->Size = System::Drawing::Size(143, 29);

this->подключениеToolStripMenuItem->Text = L"Подключение";

//

// информацияToolStripMenuItem

//

this->информацияToolStripMenuItem->Name = L"информацияToolStripMenuItem";

this->информацияToolStripMenuItem->Size = System::Drawing::Size(281, 34);

this->информацияToolStripMenuItem->Text = L"Информация";

this->информацияToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::информацияToolStripMenuItem\_Click);

//

// подключениеВБДToolStripMenuItem

//

this->подключениеВБДToolStripMenuItem->Name = L"подключениеВБДToolStripMenuItem";

this->подключениеВБДToolStripMenuItem->Size = System::Drawing::Size(281, 34);

this->подключениеВБДToolStripMenuItem->Text = L"Подключение в БД";

this->подключениеВБДToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::подключениеВБДToolStripMenuItem\_Click);

//

// закрытьПрограммуToolStripMenuItem

//

this->закрытьПрограммуToolStripMenuItem->Name = L"закрытьПрограммуToolStripMenuItem";

this->закрытьПрограммуToolStripMenuItem->Size = System::Drawing::Size(281, 34);

this->закрытьПрограммуToolStripMenuItem->Text = L"Закрыть программу";

this->закрытьПрограммуToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::закрытьПрограммуToolStripMenuItem\_Click);

//

// редактированиеToolStripMenuItem

//

this->редактированиеToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(5) {

this->новаяЗаписьToolStripMenuItem,

this->удалитьЗаписьToolStripMenuItem, this->удалитьВсеЗаписиToolStripMenuItem, this->редактированиеЗаписиToolStripMenuItem, this->удалитьТаблицуToolStripMenuItem

});

this->редактированиеToolStripMenuItem->Name = L"редактированиеToolStripMenuItem";

this->редактированиеToolStripMenuItem->Size = System::Drawing::Size(162, 29);

this->редактированиеToolStripMenuItem->Text = L"Редактирование";

//

// новаяЗаписьToolStripMenuItem

//

this->новаяЗаписьToolStripMenuItem->Name = L"новаяЗаписьToolStripMenuItem";

this->новаяЗаписьToolStripMenuItem->Size = System::Drawing::Size(308, 34);

this->новаяЗаписьToolStripMenuItem->Text = L"Новая запись";

this->новаяЗаписьToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::новаяЗаписьToolStripMenuItem\_Click);

//

// удалитьЗаписьToolStripMenuItem

//

this->удалитьЗаписьToolStripMenuItem->Name = L"удалитьЗаписьToolStripMenuItem";

this->удалитьЗаписьToolStripMenuItem->Size = System::Drawing::Size(308, 34);

this->удалитьЗаписьToolStripMenuItem->Text = L"Удалить запись";

this->удалитьЗаписьToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::удалитьЗаписьToolStripMenuItem\_Click);

//

// удалитьВсеЗаписиToolStripMenuItem

//

this->удалитьВсеЗаписиToolStripMenuItem->Name = L"удалитьВсеЗаписиToolStripMenuItem";

this->удалитьВсеЗаписиToolStripMenuItem->Size = System::Drawing::Size(308, 34);

this->удалитьВсеЗаписиToolStripMenuItem->Text = L"Удалить все записи";

this->удалитьВсеЗаписиToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::удалитьВсеЗаписиToolStripMenuItem\_Click);

//

// редактированиеЗаписиToolStripMenuItem

//

this->редактированиеЗаписиToolStripMenuItem->Name = L"редактированиеЗаписиToolStripMenuItem";

this->редактированиеЗаписиToolStripMenuItem->Size = System::Drawing::Size(308, 34);

this->редактированиеЗаписиToolStripMenuItem->Text = L"Редактирование записи";

this->редактированиеЗаписиToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::редактированиеЗаписиToolStripMenuItem\_Click);

//

// удалитьТаблицуToolStripMenuItem

//

this->удалитьТаблицуToolStripMenuItem->Name = L"удалитьТаблицуToolStripMenuItem";

this->удалитьТаблицуToolStripMenuItem->Size = System::Drawing::Size(308, 34);

this->удалитьТаблицуToolStripMenuItem->Text = L"Удалить таблицу";

this->удалитьТаблицуToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::удалитьТаблицуToolStripMenuItem\_Click);

//

// запросToolStripMenuItem

//

this->запросToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(5) {

this->выводТабToolStripMenuItem,

this->выводТаблицыПоСтатусуToolStripMenuItem, this->выводТаблицыПоСрокуСдачиToolStripMenuItem, this->выводТаблицыПоОтветственномуToolStripMenuItem,

this->пользовательскийSQLзапросToolStripMenuItem

});

this->запросToolStripMenuItem->Name = L"запросToolStripMenuItem";

this->запросToolStripMenuItem->Size = System::Drawing::Size(87, 29);

this->запросToolStripMenuItem->Text = L"Запрос";

//

// выводТабToolStripMenuItem

//

this->выводТабToolStripMenuItem->Name = L"выводТабToolStripMenuItem";

this->выводТабToolStripMenuItem->Size = System::Drawing::Size(404, 34);

this->выводТабToolStripMenuItem->Text = L"Вывод таблицы";

this->выводТабToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::выводТабToolStripMenuItem\_Click);

//

// выводТаблицыПоСтатусуToolStripMenuItem

//

this->выводТаблицыПоСтатусуToolStripMenuItem->Name = L"выводТаблицыПоСтатусуToolStripMenuItem";

this->выводТаблицыПоСтатусуToolStripMenuItem->Size = System::Drawing::Size(404, 34);

this->выводТаблицыПоСтатусуToolStripMenuItem->Text = L"Вывод таблицы по статусу";

this->выводТаблицыПоСтатусуToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::выводТаблицыПоСтатусуToolStripMenuItem\_Click);

//

// выводТаблицыПоСрокуСдачиToolStripMenuItem

//

this->выводТаблицыПоСрокуСдачиToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->вПрямомПорядкеToolStripMenuItem,

this->вОбратномПорядкеToolStripMenuItem

});

this->выводТаблицыПоСрокуСдачиToolStripMenuItem->Name = L"выводТаблицыПоСрокуСдачиToolStripMenuItem";

this->выводТаблицыПоСрокуСдачиToolStripMenuItem->Size = System::Drawing::Size(404, 34);

this->выводТаблицыПоСрокуСдачиToolStripMenuItem->Text = L"Вывод таблицы по сроку сдачи";

//

// вПрямомПорядкеToolStripMenuItem

//

this->вПрямомПорядкеToolStripMenuItem->Name = L"вПрямомПорядкеToolStripMenuItem";

this->вПрямомПорядкеToolStripMenuItem->Size = System::Drawing::Size(285, 34);

this->вПрямомПорядкеToolStripMenuItem->Text = L"В прямом порядке";

this->вПрямомПорядкеToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::вПрямомПорядкеToolStripMenuItem\_Click);

//

// вОбратномПорядкеToolStripMenuItem

//

this->вОбратномПорядкеToolStripMenuItem->Name = L"вОбратномПорядкеToolStripMenuItem";

this->вОбратномПорядкеToolStripMenuItem->Size = System::Drawing::Size(285, 34);

this->вОбратномПорядкеToolStripMenuItem->Text = L"В обратном порядке";

this->вОбратномПорядкеToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::вОбратномПорядкеToolStripMenuItem\_Click);

//

// выводТаблицыПоОтветственномуToolStripMenuItem

//

this->выводТаблицыПоОтветственномуToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ >(2) {

this->аЯToolStripMenuItem,

this->яАToolStripMenuItem

});

this->выводТаблицыПоОтветственномуToolStripMenuItem->Name = L"выводТаблицыПоОтветственномуToolStripMenuItem";

this->выводТаблицыПоОтветственномуToolStripMenuItem->Size = System::Drawing::Size(404, 34);

this->выводТаблицыПоОтветственномуToolStripMenuItem->Text = L"Вывод таблицы по ответственному";

//

// аЯToolStripMenuItem

//

this->аЯToolStripMenuItem->Name = L"аЯToolStripMenuItem";

this->аЯToolStripMenuItem->Size = System::Drawing::Size(154, 34);

this->аЯToolStripMenuItem->Text = L"А - Я";

this->аЯToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::аЯToolStripMenuItem\_Click);

//

// яАToolStripMenuItem

//

this->яАToolStripMenuItem->Name = L"яАToolStripMenuItem";

this->яАToolStripMenuItem->Size = System::Drawing::Size(154, 34);

this->яАToolStripMenuItem->Text = L"Я - А";

this->яАToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::яАToolStripMenuItem\_Click);

//

// пользовательскийSQLзапросToolStripMenuItem

//

this->пользовательскийSQLзапросToolStripMenuItem->Name = L"пользовательскийSQLзапросToolStripMenuItem";

this->пользовательскийSQLзапросToolStripMenuItem->Size = System::Drawing::Size(404, 34);

this->пользовательскийSQLзапросToolStripMenuItem->Text = L"Пользовательский SQL-запрос";

this->пользовательскийSQLзапросToolStripMenuItem->Click += gcnew System::EventHandler(this, &MainWindow::пользовательскийSQLзапросToolStripMenuItem\_Click);

//

// выводВФайлToolStripMenuItem1

//

this->выводВФайлToolStripMenuItem1->Name = L"выводВФайлToolStripMenuItem1";

this->выводВФайлToolStripMenuItem1->Size = System::Drawing::Size(142, 29);

this->выводВФайлToolStripMenuItem1->Text = L"Вывод в файл";

this->выводВФайлToolStripMenuItem1->Click += gcnew System::EventHandler(this, &MainWindow::выводВФайлToolStripMenuItem\_Click);

//

// выводВФайлToolStripMenuItem

//

this->выводВФайлToolStripMenuItem->Name = L"выводВФайлToolStripMenuItem";

this->выводВФайлToolStripMenuItem->Size = System::Drawing::Size(32, 19);

//

// dataGridView1

//

this->dataGridView1->AllowUserToAddRows = false;

this->dataGridView1->AllowUserToDeleteRows = false;

this->dataGridView1->AutoSizeColumnsMode = System::Windows::Forms::DataGridViewAutoSizeColumnsMode::Fill;

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(6) {

this->Column1,

this->Column2, this->Column3, this->Column4, this->Column5, this->Column6

});

this->dataGridView1->Dock = System::Windows::Forms::DockStyle::Fill;

this->dataGridView1->Location = System::Drawing::Point(0, 33);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->ReadOnly = true;

this->dataGridView1->RowHeadersWidth = 40;

this->dataGridView1->RowTemplate->Height = 28;

this->dataGridView1->Size = System::Drawing::Size(1898, 991);

this->dataGridView1->TabIndex = 16;

//

// Column1

//

this->Column1->FillWeight = 20;

this->Column1->HeaderText = L"ID";

this->Column1->MinimumWidth = 8;

this->Column1->Name = L"Column1";

this->Column1->ReadOnly = true;

//

// Column2

//

this->Column2->HeaderText = L"Название";

this->Column2->MinimumWidth = 8;

this->Column2->Name = L"Column2";

this->Column2->ReadOnly = true;

//

// Column3

//

this->Column3->HeaderText = L"Описание";

this->Column3->MinimumWidth = 8;

this->Column3->Name = L"Column3";

this->Column3->ReadOnly = true;

//

// Column4

//

this->Column4->HeaderText = L"Статус";

this->Column4->MinimumWidth = 8;

this->Column4->Name = L"Column4";

this->Column4->ReadOnly = true;

//

// Column5

//

this->Column5->HeaderText = L"Ответственный";

this->Column5->MinimumWidth = 8;

this->Column5->Name = L"Column5";

this->Column5->ReadOnly = true;

//

// Column6

//

this->Column6->HeaderText = L"Срок сдачи";

this->Column6->MinimumWidth = 8;

this->Column6->Name = L"Column6";

this->Column6->ReadOnly = true;

//

// MainWindow

//

this->AutoScaleDimensions = System::Drawing::SizeF(9, 20);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->ClientSize = System::Drawing::Size(1898, 1024);

this->Controls->Add(this->dataGridView1);

this->Controls->Add(this->menuStrip1);

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->MainMenuStrip = this->menuStrip1;

this->MaximumSize = System::Drawing::Size(1920, 1080);

this->MinimumSize = System::Drawing::Size(640, 360);

this->Name = L"MainWindow";

this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;

this->Text = L"DBROM";

this->WindowState = System::Windows::Forms::FormWindowState::Maximized;

this->menuStrip1->ResumeLayout(false);

this->menuStrip1->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

//

// Подключение к БД

//

private: void connect\_to\_DB()

{

SqlConnectionStringBuilder^ strbild = gcnew SqlConnectionStringBuilder();

strbild->DataSource = data\_source;

strbild->InitialCatalog = initial\_catalog;

strbild->IntegratedSecurity = true;

connection = gcnew SqlConnection(Convert::ToString(strbild));

}

//

// Процедура создания новой записи

//

private: void insert\_data(String^ label, String^ description, String^ status, String^ admin, String^ deadline)

{

connect\_to\_DB();

String^ command\_text = "INSERT INTO " + t\_name +"(label, description, status, admin, deadline) VALUES(@Label, @Description, @Status, @Admin, @Deadline)";

SqlCommand^ cmd = gcnew SqlCommand(command\_text, connection);

cmd->Parameters->AddWithValue("@Label", label);

cmd->Parameters->AddWithValue("@Description", description);

cmd->Parameters->AddWithValue("@Status", status);

cmd->Parameters->AddWithValue("@Admin", admin);

cmd->Parameters->AddWithValue("@Deadline", Convert::ToDateTime(deadline));

connection->Open();

cmd->ExecuteNonQuery();

connection->Close();

output\_table(dataGridView1, "SELECT \* FROM " + t\_name);

}

//

// Процедура чтения одного ряда из таблицы запроса

//

private: void read\_single\_row(DataGridView^ dgw, IDataRecord^ record)

{

dgw->Rows->Add(record->GetInt32(0), record->GetString(1), record->GetString(2), record->GetString(3), record->GetString(4), Convert::ToString(record->GetDateTime(5))->Remove(10));

}

//

// Процедура вывода полученной таблицы

//

private: void output\_table(DataGridView^ dgw, String^ query)

{

prev\_query = query;

dgw->Rows->Clear();

connect\_to\_DB();

SqlCommand^ cmd = gcnew SqlCommand(query, connection);

connection->Open();

SqlDataReader^ reader = cmd->ExecuteReader();

while (reader->Read())

{

read\_single\_row(dgw, reader);

}

reader->Close();

connection->Close();

}

//

// Процедура запускающая действия при нажатии "Новая запись"

//

private: System::Void новаяЗаписьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

NewRecord^ nr = gcnew NewRecord();

nr->ShowDialog(this);

if (nr->get\_label() != "" && nr->get\_description() != "" && nr->get\_status() != "" && nr->get\_admin() != "" && Convert::ToString(nr->get\_deadline()) != "")

{

insert\_data(nr->get\_label(), nr->get\_description(), nr->get\_status(), nr->get\_admin(), Convert::ToString(nr->get\_deadline()));

}

}

//

// Процедура запускающая действия при нажатии "Удалить все записи"

//

private: System::Void удалитьВсеЗаписиToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Вы уверены, что хотите удалить все записи?", "Уведомление", MessageBoxButtons::YesNo, MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes)

{

connect\_to\_DB();

String^ command\_text = "DELETE " + t\_name;

SqlCommand^ cmd = gcnew SqlCommand(command\_text, connection);

connection->Open();

cmd->ExecuteNonQuery();

connection->Close();

}

}

//

// Процедура запускающая действия при нажатии "Вывод таблицы"

//

private: System::Void выводТабToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

output\_table(dataGridView1, "SELECT \* FROM " + t\_name);

}

//

// Процедура запускающая действия при нажатии "Вывод таблицы по статусу"

//

private: System::Void выводТаблицыПоСтатусуToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

output\_table(dataGridView1, "SELECT \* FROM " + t\_name + " ORDER BY status");

}

//

// Процедура запускающая действия при нажатии "А - Я"

//

private: System::Void аЯToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

output\_table(dataGridView1, "SELECT \* FROM " + t\_name + " ORDER BY admin ASC"); /\*id, admin, label, description, status, deadline FROM\*/

}

//

// Процедура запускающая действия при нажатии "Я - А"

//

private: System::Void яАToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

output\_table(dataGridView1, "SELECT \* FROM " + t\_name + " ORDER BY admin DESC"); /\*id, admin, label, description, status, deadline FROM\*/

}

//

// Процедура запускающая действия при нажатии "В прямом порядке"

//

private: System::Void вПрямомПорядкеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

output\_table(dataGridView1, "SELECT \* FROM " + t\_name + " ORDER BY deadline ASC");

}

//

// Процедура запускающая действия при нажатии "В обратном порядке"

//

private: System::Void вОбратномПорядкеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

output\_table(dataGridView1, "SELECT \* FROM " + t\_name + " ORDER BY deadline DESC");

}

//

// Процедура запускающая действия при нажатии "Удалить запись"

//

private: System::Void удалитьЗаписьToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

DeleteRecord^ dr = gcnew DeleteRecord(data\_source, initial\_catalog, t\_name);

dr->ShowDialog(this);

output\_table(dataGridView1, "SELECT \* FROM " + t\_name);

}

//

// Процедура запускающая действия при нажатии "Редактирование записи"

//

private: System::Void редактированиеЗаписиToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

EditRecord^ er = gcnew EditRecord(data\_source, initial\_catalog, t\_name);

er->ShowDialog(this);

output\_table(dataGridView1, "SELECT \* FROM " + t\_name);

}

//

// Процедура запускающая действия при нажатии "Информация"

//

private: System::Void информацияToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Для работы программы необходимо подключиться к SQL серверу. На сервере необходимо создать базу дынных и таблицу с произвольными названиями.\r\n\r\nДля корректной работы программы таблица базы данных должна удовлетворять следующим условиям:\r\n1. У колонок должны быть следующие названия, сначала идет имя колонки, в скобках тип данных: ID (int), label (varchar(200)), description (varchar(200)), status (varchar(200)), admin (varchar(200)), deadline (date)\r\n2. У колонки ID должен быть включён параметр \"Identity Specification\"\r\n3. У всех колонок должен быть отключён параметр \"Allow Nulls\"\r\n\r\nДля подключения к созданной БД выберите \"Подключение к БД\", в открывшемся окне введите название сервера и базы данных", "Информация о подключении к локальному SQL серверу", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

//

// Процедура подключения к БД при вызове конструктора

//

private: void connection\_to\_constr()

{

try

{

ConnectionWindow^ cwc = gcnew ConnectionWindow(false);

cwc->ShowDialog(this);

data\_source = cwc->get\_DS();

initial\_catalog = cwc->get\_IC();

t\_name = cwc->get\_table\_name();

serviceName = cw->get\_serv\_name();

if (cwc->ex == "1")

{

this->Close();

}

else

{

if (data\_source == "" && initial\_catalog == "" && this->t\_name == "") this->Close();

else

{

output\_table(dataGridView1, "SELECT \* FROM " + t\_name);

if (data\_source != "" && initial\_catalog != "" && t\_name != "")

{

if (this->Text != "DBROM") this->Text = this->Text->Remove(5) + " — " + t\_name;

else this->Text = this->Text + " — " + t\_name;

}

}

}

delete cwc;

}

catch (System::ObjectDisposedException^) { this->Close(); }

}

//

// Процедура подключения в БД

//

private: void connection\_to()

{

ConnectionWindow^ cw = gcnew ConnectionWindow(true);

cw->ShowDialog(this);

data\_source = cw->get\_DS();

initial\_catalog = cw->get\_IC();

t\_name = cw->get\_table\_name();

serviceName = cw->get\_serv\_name();

if(cw->ex != "1")

{

if (this->Text != "DBROM") this->Text = this->Text->Remove(5) + " — " + t\_name;

else this->Text = this->Text + " — " + t\_name;

}

}

//

// Процедура запускающая действия при нажатии "Подключение к БД"

//

private: System::Void подключениеВБДToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

connection\_to();

if(t\_name != "")

output\_table(dataGridView1, "SELECT \* FROM " + t\_name);

}

//

// Процедура запускающая действия при нажатии "Удалить таблицу"

//

private: System::Void удалитьТаблицуToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

System::Windows::Forms::DialogResult result = MessageBox::Show("Вы уверены, что хотите удалить эту таблицу?", "Уведомление", MessageBoxButtons::YesNo, MessageBoxIcon::Question);

if (result == System::Windows::Forms::DialogResult::Yes)

{

dataGridView1->Rows->Clear();

connect\_to\_DB();

String^ query = "DROP TABLE " + t\_name;

SqlCommand^ cmd = gcnew SqlCommand(query, connection);

connection->Open();

cmd->ExecuteNonQuery();

connection->Close();

this->Text = "DBROM";

подключениеВБДToolStripMenuItem\_Click(sender, e);

}

}

//

//

//

private: System::Void выводВФайлToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

String^ path;

if (SaveFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)

{

path = SaveFileDialog->FileName;

if (path != "")

{

StreamWriter^ writer = gcnew StreamWriter(path, false);

writer->Close();

}

}

if (path != nullptr)

{

String^ data = "";

connect\_to\_DB();

SqlCommand^ cmd = gcnew SqlCommand(prev\_query, connection);

connection->Open();

SqlDataReader^ reader = cmd->ExecuteReader();

while (reader->Read())

{

data += "ID записи:\t" + reader->GetInt32(0) + "\r\nНазвание:\t" + reader->GetString(1) + "\r\nОписание:\t" + reader->GetString(2) + "\r\nСтатус:\t\t" + reader->GetString(3) + "\r\nОтветсвенный:\t" + reader->GetString(4) + "\r\nСрок сдачи:\t" + Convert::ToString(reader->GetDateTime(5))->Remove(10) + "\r\n\r\n";

}

reader->Close();

connection->Close();

StreamWriter^ writer = gcnew StreamWriter(path);

writer->Write(data);

writer->Close();

MessageBox::Show("Данных сохранены в файл по адресу: " + path, "Уведомление", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

}

//

// Процедура запускающая действия при нажатии "Пользовательский SQL-запрос"

//

private: System::Void пользовательскийSQLзапросToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

{

connect\_to\_DB();

String^ query = "SELECT TOP(1000)[ID], [label], [description], [status], [admin], [deadline] FROM[" + t\_name + "]";

SqlCommand^ cmd = gcnew SqlCommand(query, connection);

connection->Open();

cmd->ExecuteNonQuery();

connection->Close();

}

int x = this->Left + (this->Width / 2) - 200;

int y = this->Top + (this->Height / 2) - 100;

int check = 0;

String^ pr\_q = prev\_query;

String^ query = "";

query = Interaction::InputBox("Введите любой SQL-запрос, за исключением запроса на выборку (в этой функции данные не выводятся на экран)", "SQL-запрос", query, x, y);

try

{

//output\_table(dataGridView1, query);

connect\_to\_DB();

SqlCommand^ cmd = gcnew SqlCommand(query, connection);

connection->Open();

cmd->ExecuteNonQuery();

connection->Close();

MessageBox::Show("SQL-запрос выполнен", "Уведомление", MessageBoxButtons::OK, MessageBoxIcon::Information);

}

catch (System::Exception^)

{

MessageBox::Show("Введён некорректный SQL-запрос", "Ошибка", MessageBoxButtons::OK, MessageBoxIcon::Error);

prev\_query = pr\_q;

}

output\_table(dataGridView1, "SELECT TOP(1000)[ID], [label], [description], [status], [admin], [deadline] FROM[" + t\_name + "]");

}

private: System::Void закрытьПрограммуToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

this->Close();

}

};

}