Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСИС».

Институт компьютерных наук.

Кафедра инженерной кибернетики.

Домашнее задание по курсу «Объектно-ориентированное программирование»

на тему:

«Система планирования «Ежедневник»»

Выполнила: Игнатова Мария Евгеньевна

Студент группы БИВТ-23-10

Научный руководитель:

Куренкова Татьяна Васильевна

Москва 2024

**Содержание**

1. Цель работы………………………………………………………….3
2. Основная часть………………………………………………...…4-17
   1. Техническое задание………………………………………………4-5
   2. Описание интерфейса……………………………………………5-10
   3. Техническое описание………………………………………….10-17
3. Тестирование……………………………………………………….18
4. Документирование…………………………………………………19

Заключение………………………………………………………………..20

Список использованной литературы…………………………………….21

Приложение №1……………………………………………………….21-27

Приложение №2……………………………………………………….28-39

**1 Цель работы**

Цель работы: разработать приложение, позволяющее эффективно использовать ресурс времени за счет структурирования задач, сортировки их по важности, наглядного отображения всего списка дел.

**2 Основная часть**

2.1 Техническое задание

Требования к функциональным характеристикам: Система должна обеспечивать возможность выполнения следующих функций:

* 1. Регистрация и аутентификация пользователей.
  2. Создание, редактирование и удаление записей о задачах на определенную дату.
  3. Возможность добавления приоритета, сроков выполнения к задачам.
  4. Функция поиска задач.
  5. Сортировка списка задач по заданному параметру: приоритет, срок выполнения, наименование задачи.
  6. Отображение графика загруженности предстоящей недели.
  7. Определение количества задач, срок выполнения которых истек.

Исходные данные:

1. Списки пользователей.
2. Сведения о текущих задачах конкретного пользователя.
3. Сведения о сроках выполнения и приоритете конкретной задачи.

Результат:

1. Отсортированный по критериям (приоритет, срок выполнения, наименование задачи) список задач.
2. Отображённый график загруженности предстоящей недели.
3. Отображение самого загруженного дня предстоящей недели и количества просроченных задач на текущий момент.
4. Отображение списка задач, подходящих по критерию поиска.

Диаграмма взаимодействия пользователя с системой (рис. 1).

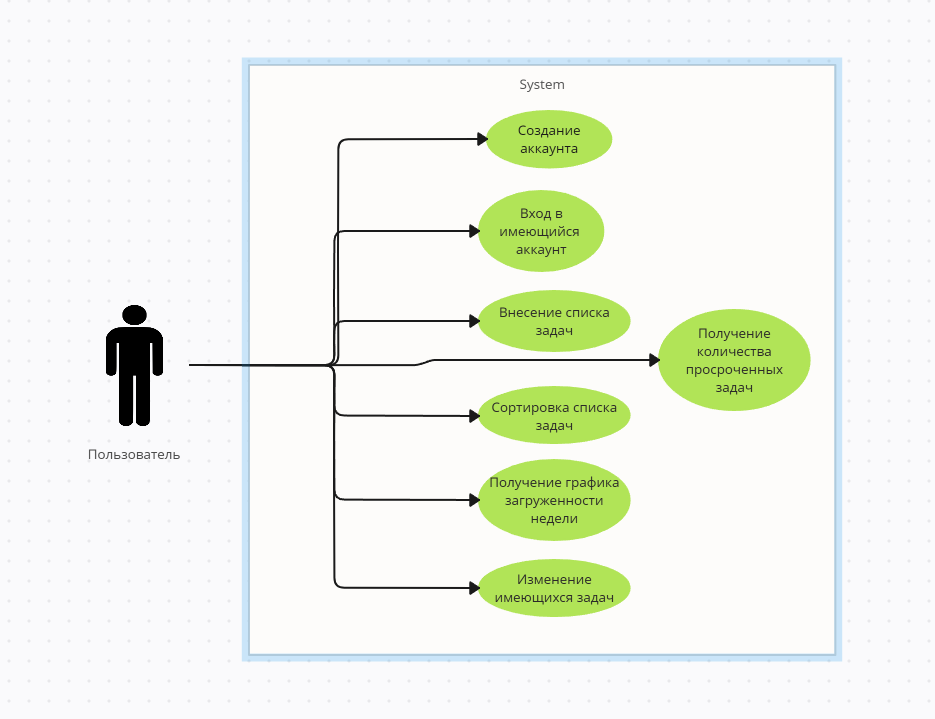


Рис.1

2.2 Описание интерфейса

Пользователю доступно 4 формы для взаимодействия с системой: форма авторизации (рис. 2), форма регистрации (рис. 3), основная форма «Ежедневника» (рис. 4.1 – 4.2), форма добавления новой задачи (рис. 5). Основная форма имеет 2 вкладки: вкладка «Список задач» (рис 4.1), вкладка «График загруженности» (рис. 4.2).

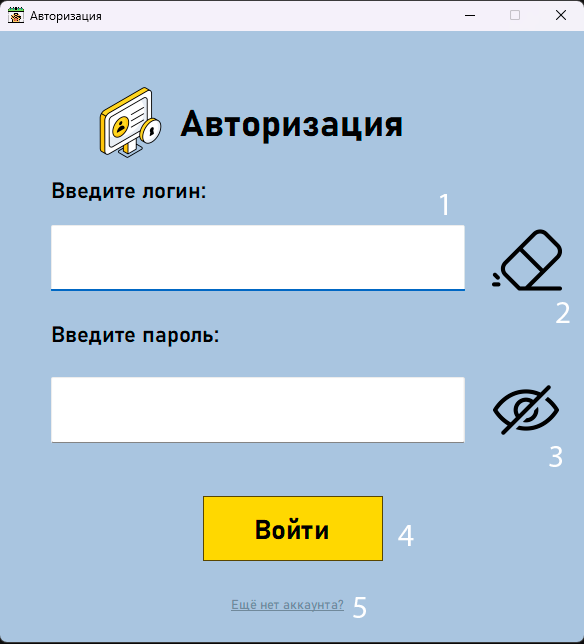
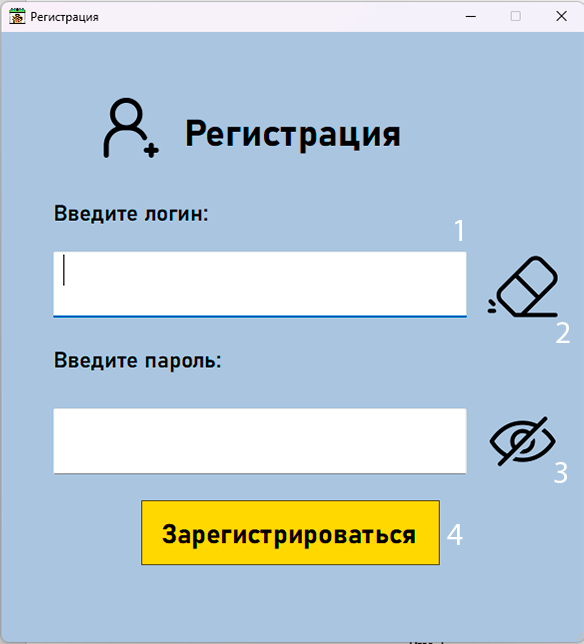


Рис. 2 Рис. 3

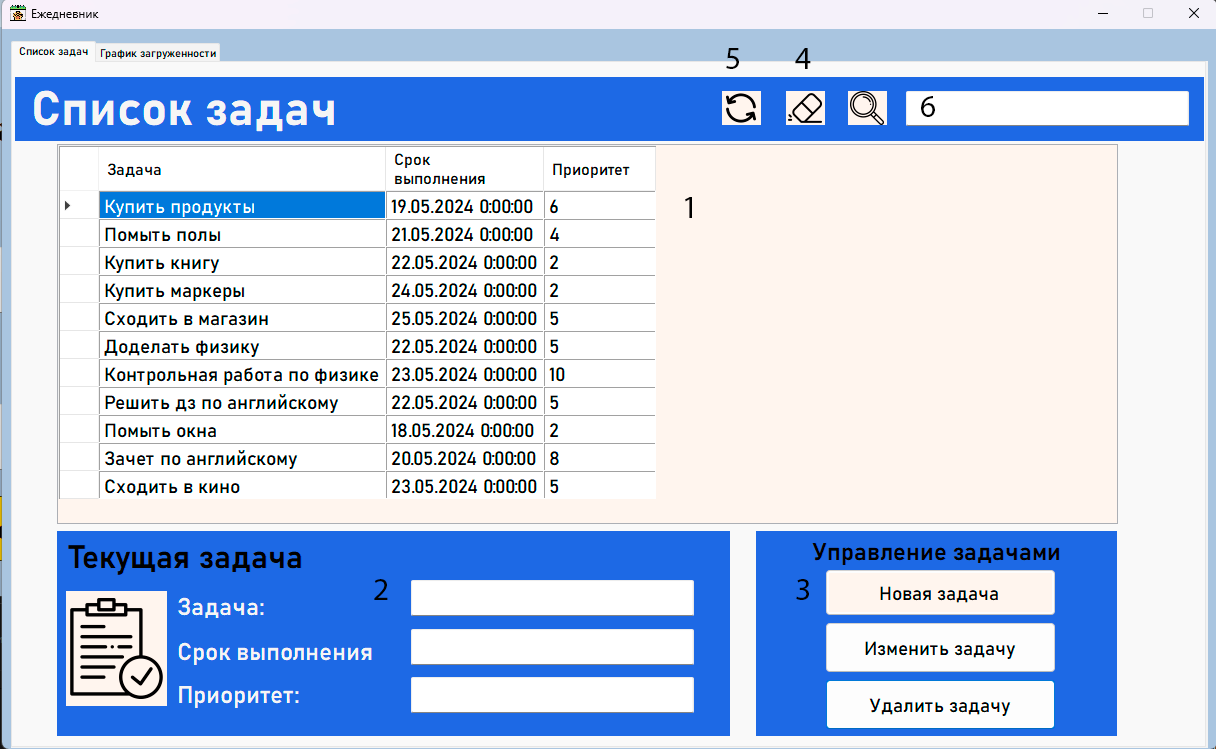


Рис. 4.1

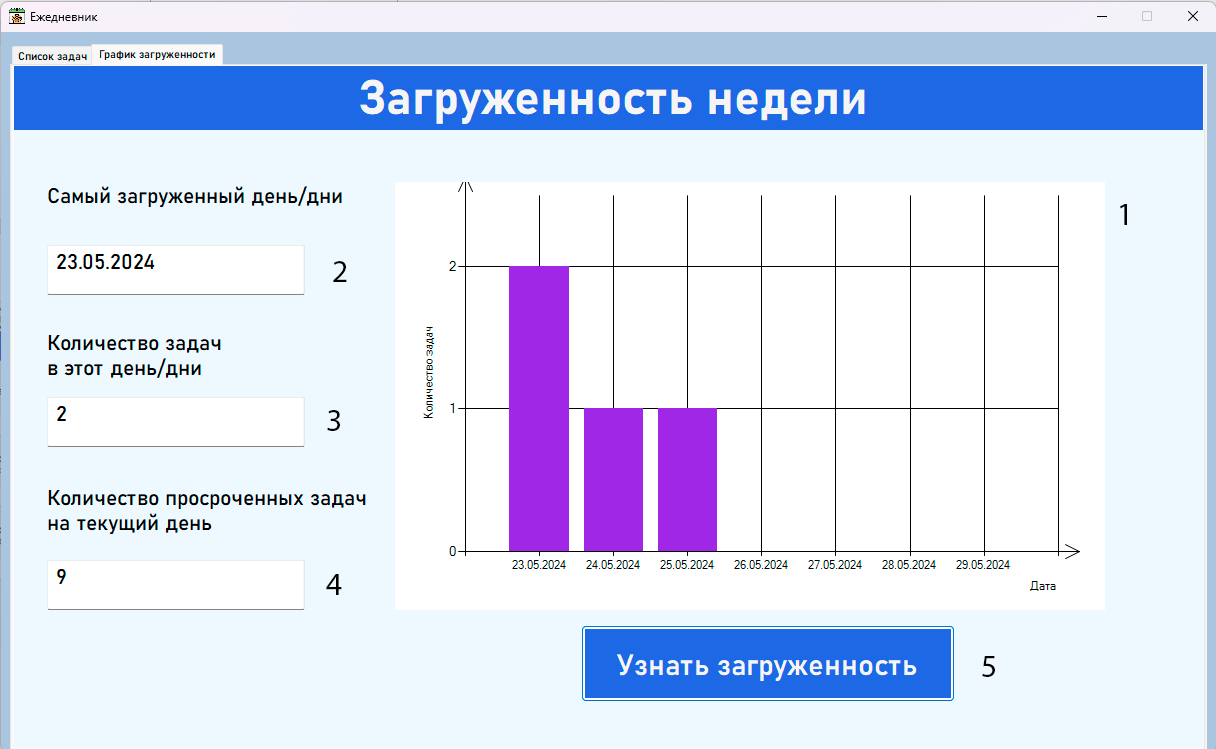


Рис. 4.2

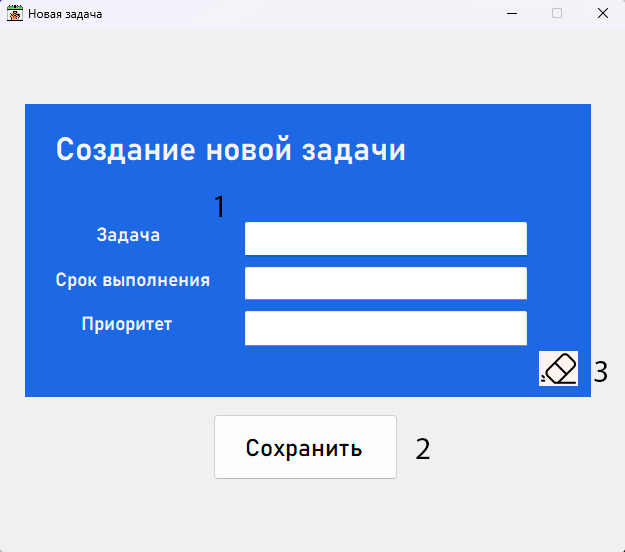


Рис. 5

При открытии приложения пользователь попадает на форму авторизации (рис. 2), на которой расположены:

1. Поля для ввода логина и пароля.
2. Кнопка очистки полей ввода.
3. Кнопка отображения пароля.
4. Кнопка «Войти» входа в систему.
5. Текст-ссылка «Ещё нет аккаунта?» для перехода на форму регистрации

Если у пользователя имеется аккаунт в системе, он вводит свои данные, нажимает кнопку «Войти» и входит в систему.

Если пользователь еще не авторизован в системе, он нажимает на текст-ссылку «Ещё нет аккаунта?» внизу текущей формы и попадает на форму регистрации (рис. 3).

На форме регистрации расположены:

1. Поля для ввода логина и пароля.
2. Кнопка очистки полей ввода.
3. Кнопка отображения пароля.
4. Кнопка «Регистрация» для создания нового аккаунта.

Для регистрации в приложении пользователь вводит в отведённые поля свои логин и пароль и нажимает кнопку «Регистрация». Открывается форма авторизации для входа в систему. Пользователь выполняет последовательность действий для входа, описанную выше.

При попадании на основную форму «Ежедневника» (рис. 4.1-4.2) пользователю доступны две вкладки, описанные выше.

На вкладке «Список задач» расположены:

1. Поле для отображения списка задач.
2. Поля для отображения информации о конкретной задачи.
3. Кнопки управления задачами («Новая задача», «Изменить задачу», «Удалить задачу»).
4. Кнопка для очистки всех полей ввода.
5. Кнопка для обновления списка задач.
6. Поле для ввода текста для поиска задачи по конкретному параметру.

Для добавления новой задачи пользователь нажимает на кнопку «Новая задача» и переходит на форму добавления новой задачи (рис. 5).

На форме добавления новой задачи расположены:

1. Поля для ввода информации о новой задаче
2. Кнопка «Сохранить» для добавления задачи в список
3. Кнопка очистки всех полей ввода.

Пользователь вводит данные о необходимой ему задаче и нажимает кнопку «Сохранить». Задача добавляется в список. Последовательность действий повторяется необходимое количество раз. После завершении добавления новых задач пользователь закрывает форму добавления новых задач.

Для отображения добавленных задач пользователю необходимо нажать на кнопку обновления списка задач (рис 4.1, 5)

При нажатии на любую задачу информация о ней переносится в поля для отображения информации о данной задаче автоматически. Для изменения какого-либо параметра задачи пользователь вводит новые значения в указанные выше поля, после чего нажимает кнопку «Изменить задачу».

Для удаления задачи пользователю необходимо выбрать строку с нужной задачей, нажав на крайний левый столбец в строке выбранной задачи, затем нажать кнопку «Удалить задачу».

Для обновления списка задач пользователю необходимо нажать на кнопку обновления списка задач (рис. 4.1, 5).

Для очистки все полей ввода пользователю необходимо нажать на кнопку очистки всех полей ввода (рис. 4.1, 4).

Для поиска задачи по заданному параметру (приоритет, срок выполнения, наименование задачи), пользователю необходимо начать вводить запрос в поле для ввода текста для поиска задачи по конкретному параметру. Задачи, совпадающие с заданным критерием, выводятся в список автоматически. Дополнительно нажимать кнопки не требуется.

Для сортировки списка задач пользователю необходимо выбрать нужный параметр сортировки и нажать однократно на ячейку с названием столбца. Список будет отсортирован по возрастанию. При повторном нажатии сортировка будет по убыванию. Для отмены сортировки необходимо обновить список задач (см. выше).

На вкладке «График загруженности» расположены:

1. Поле для вывода графика загруженности предстоящей недели
2. Поле для вывода самого загруженного дня
3. Поле для вывода количества задач в самый загруженный
4. Поле для вывода количества просроченных задач
5. Кнопка «Узнать загруженность»

Для отображения информации о загруженности предстоящей недели (считая настоящий момент времени начальным) пользователю необходимо нажать на кнопку «Узнать загруженность». Искомая информация отобразится на графике и указанных выше полях.

Для выхода из приложения пользователю необходимо нажать на красный крестик в верхнем правом углу основной формы «Ежедневника».

2.3 Техническое описание

Используемые ресурсы:

1. Язык программирования: C#

2. GUI фреймворк: Windows Forms

3. База данных: SQLite

Структура базы данных SQLite «diary» (рис. 6) для хранения данных о задачах:

- Таблица "Users"с полями:

- "id" INTEGER

- "login\_user" TEXT,

- "password\_user" TEXT

- PRIMARY KEY("id" AUTOINCREMENT)

- Таблица "Users\_Task" с полями:

- "task\_id" INTEGER NOT NULL

- "task" TEXT

- "deadline" TEXT

- "priority" INTEGER

- "user\_id" INTEGER NOT NULL

- PRIMARY KEY("task\_id" AUTOINCREMENT)

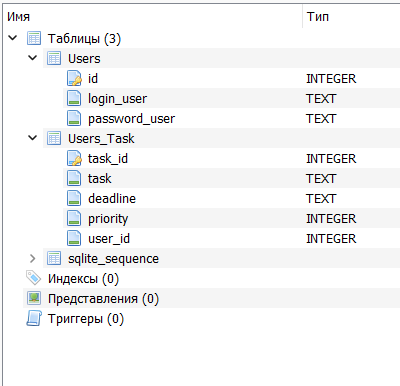
****

Рис. 6

Таблица «Users» хранит сведения о всех зарегистрированных пользователях.

Таблица «Users\_Task» хранит сведения о задачах всех зарегистрированных пользователях.

Рассмотрим коды классов продукта.

Класс Users.cs (см. приложение №1, п. 2) представляет данные пользователя (номер в системе, логин и пароль пользователя), хранимые в полях int id, string login\_user, string password\_user.

В классе содержаться три метода, необходимые для взаимодействия пользователя и форм авторизации и регистрации.

Метод CheckingUser() (см. приложение №1, п. 2) проверяет, правильно ли введены логин и пароль уже зарегистрированного пользователя. Возвращает логические значения true или false. Реализуется на форме авторизации.

Метод ExistenceUser() (см. приложение №1, п. 2) проверяет, существует ли в базе данных пользователь с введённым в поле логином. Возвращает логические значения true или false. Реализуется на форме регистрации.

Метод UpdateUser() (см. приложение №1, п. 2) добавляет нового пользователя в базу данных. Данные о логине и пароле берёт из соответствующих полей на форме регистрации.

Класс Users\_Task.cs (см. приложение №1, п. 2) представляет данные о задачах конкретного пользователя (номер задач, наименование задачи, срок выполнения, приоритет, номер пользователя), хранимые в полях int task\_id, string task, DateTime deadline, int priority, int user\_id.

Класс позволяет создать экземпляры класса, которые затем помещаются в List<Users\_Task> listname (пример), необходимый для отображения списка задач.

Поэтому в классе содержится статический метод ReadFromDB (int user\_id) (см. приложение №1, п. 2), получающий на вход значение номера пользователя в системе, который создаёт экземпляр класса List<Users\_Task> tasks, вносит в него подходящие по критерию (user\_id) значения и возвращает его.

Класс реализуется на основной форме приложения (см. приложение №2 class Plan.cs).

Класс Users.cs и Users\_Task.cs связаны между собой значением номера пользователя в системе, который описывается полями int id и int user\_id соответственно. Значение номера пользователя передаётся между формами следующим образом (рис. 7 и рис. 8):

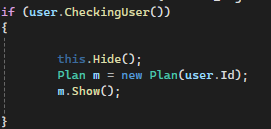
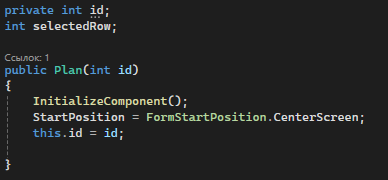


Рис. 7 Рис. 8

Это позволяет управлять информацией о задачах конкретного пользователя, выводить список задач, которые принадлежат конкретному пользователю.

Рассмотрим коды форм продукта.

Первоначально пользователь попадает на форму авторизации, её интерфейс рассмотрен выше.

В классе Log\_in.cs в начальный момент времени происходит инициализация формы и установка её положения на экране.

Код этой формы выполняет вход в приложение уже авторизованного пользователя или переход на форму регистрации.

При нажатии на кнопку «Войти» в методе buttonEnter\_Click(object sender, EventArgs e) (см. приложение №2) создаётся экземпляр класса Users user. Далее с помощью метода CheckingUser() (см. выше) определяется корректность введённых данных. В случае правильного ввода логина и пароля пользователь переходит на основную форму приложения. В противном случае пользователь нажимает на текст-ссылку внизу формы авторизации и переходит на форму регистрации.

В классе Sign\_up.cs происходит внесение нового пользователя в базу данных при нажатии на кнопку «Зарегистрироваться» методом UpdateUser() (см. выше). При этом выполняется проверка на наличие пользователя с таким логином в системе методом ExistenceUser() (см. выше). В случае успешной регистрации вновь откроется форма авторизации для входа в систему.

В обоих классах добавлены методы, которые при нажатии на кнопку очищают все поля ввода и отображают/скрывают пароль (см. приложение №2).

Класс Plan.cs реализует основное взаимодействие пользователя с приложением.

Класс принимает значение номера пользователя в базе данных, переданное из предыдущей формы авторизации (см. рис. 7 и рис. 8).

В классе содержатся методы, используемые для отображения и обновления списка дел, поиска, сортировки, добавления и изменения задач, отображения графика и сопутствующих данных.

Метод CreateColumns() (см. приложение №2) создает и обозначает столбцы для DataGridView, расположенного на вкладке «Список задач».

Метод RefreshDataGridView(DataGridView dgw) (см. приложение №2) записывает список задач в DataGridView, то есть обновляет его. При этом используется метод ReadFromDB (int user\_id) (см. выше), который определяет значение локальной переменной List <Users\_Task> Tasks, данные из которой передаются в DataGridView.

Метод Search (DataGridView dgw) (см. приложение №2) обеспечивает поиск задачи в списке по заданному критерию. Он передаёт запрос к базе данных на совпадение информации из поля поиска с информацией в таблице Users\_Task. Все совпадения выводит в DataGridView.

Метод ClearTextBox() (см. приложение №2)очищает все поля ввода, присваивая им значение пустой строки.

Метод DeleteSelectedRow(int user\_id) (см. приложение №2) удаляет выбранную задачу из списка. При этом он проверяет, выбрана ли строка, в противном случае выводит сообщение с просьбой выбрать нужную строку. При корректном выборе строки метод создает запрос к базе данных для удаления строки с конкретным значение номера задачи. Затем удаляет выбранную строку из DataGridView и выводит сообщение об успешном удалении задачи.

Метод ChangeData() (см. приложение №2) позволяет изменять задачи в списке дел. Пользователь нажимает на нужную задачу, и информация о ней отображается в полях снизу. Далее он меняет информацию на ту, которая ему необходима и нажимает кнопку сохранить. Метод же сначала проверяет корректность введённых данных, выводя сообщения об ошибке при надобности, затем меняет информацию на нужную в DataGridView. После создаёт запрос к базе данных для изменения данных в ней на необходимые, указывая номер задачи и номер пользователя для корректности изменения данных (т.к. у разных пользователей могут быть одинаковые задачи, а автоинкремент у task\_id и user\_id поможет избежать путаницы).

Метод GetCountTasks(List<Users\_Task> tasks) (см. приложение №2) позволяет получить загруженность предстоящей недели. В ней мы создаем экземпляр класса Dictionary<string, int> tasksPerDay, в котором будем хранить дни и количества задач. Далее определяем рамки предстоящей недели, даобавляя к значению времени в настоящий момент 7 дней. Далее перебираем даты в этом диапазоне, проверяя количества задач в каждый из дней с помощью запросов к List<Users\_Task> tasks, который принимается на вход методом. С помощью ранее созданного словаря создаём строку, содержащую все дни с максимальным количеством задач. Также определяется значение количества пропущенных задач на текущий момент времени.

Метод dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e) (см. приложение №2) записывает данные о выделенной задаче в поля, расположенные в нижней части формы(см. выше «Описание интерфейса»).

Метод Plan\_FormClosing(object sender, FormClosingEventArgs e) (см. приложение №2) полностью закрывает приложение при нажатии на кнопку закрытия основной формы «Ежедневника» (см. выше «Описание интерфейса»).

Также в классе содержаться методы, исполняемые при нажатии на соответствующие кнопки (см. приложение №2), которые реализуют описанные выше методы. Например, при нажатии на кнопку очистки полей ввода вызывается метод ClearTextBox().

При нажатии на кнопку «Новая задача» открывается форма добавления новой задачи (см. выше «Описание интерфейса»).

Класс Newtask.cs реализует добавление пользователем новой задачи в список задач и базу данных соответственно.

Класс принимает значение номера пользователя в базе данных, переданное из предыдущей основной формы «Ежедневника» (см. рис. 7 и рис. 8).

При нажатии на кнопку «Сохранить» выполняется метод button\_save\_Click(object sender, EventArgs e) (см. приложение №2) , который создаёт запрос к базе данных и добавляет в неё новые данные с уточнением переданного ранее значения номера пользователя. Также происходит проверка на корректность введённых значений срока исполнения и приоритета.

Определён метод для очистки полей при нажатии на кнопку очистки (см. приложение №2).

Диаграмма классов приложения «Ежедневник» представлена на рисунке 8.

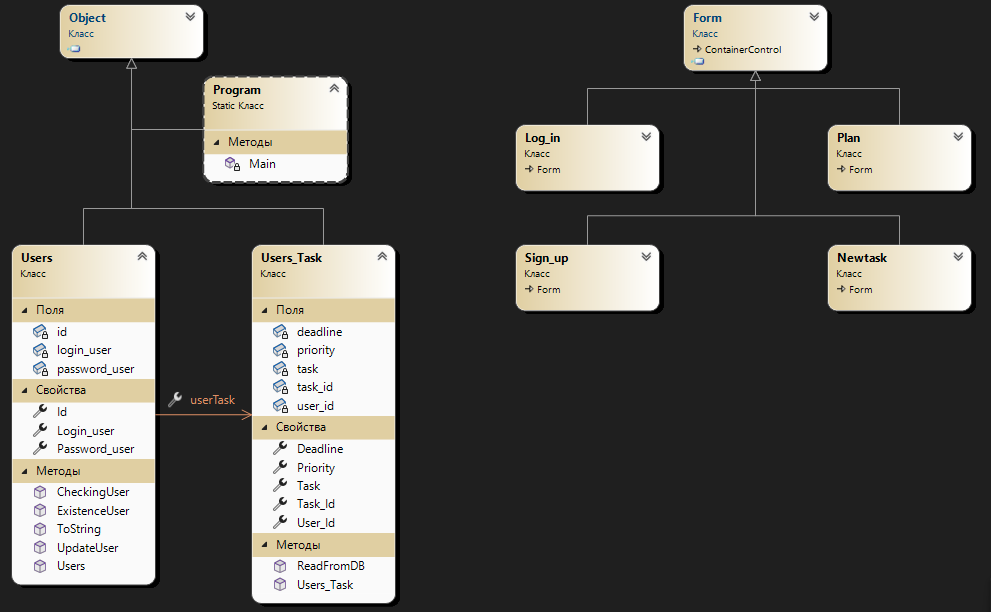


Рис. 8

**3 Тестирование**

Проведено тестирование метода ReadFromDB (int user\_id) (см. приложение №1, п. 2), содержащегося в классе Users\_task (см. приложение №1). Для этого создан модульный тест, который инициализирует массив и заполняет его ожидаемыми данными, создает массив с помощью метода ReadFromDB (int user\_id) и сравнивает результаты. Также для тестов создана тестовая база данных SQLite TestDB (рис. 9) с заданными исходными данными, равных данным в массиве с ожидаемыми результатами.

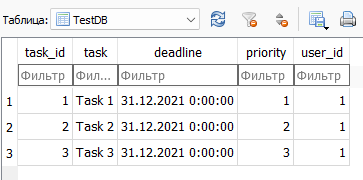


Рис. 9

Результат запуска всех тестов показан на рисунке 10 – успешен.

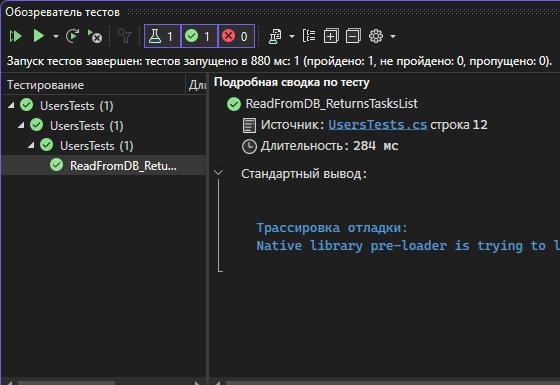


Рис. 10

(Код модульных тестов смотри в приложении №1 п. 4 и п.5).

**4 Документирование**

Все программные модули самодокументированы, текст кода смотри в приложении 1 и приложении 2.

**Заключение**

Разработка приложения на C# для системы планирования "Ежедневник" является интересным и полезным проектом. Создание такого приложения позволит пользователям эффективно организовывать свой рабочий и личный график, управлять задачами, встречами и планами.

При разработке приложения необходимо уделить особое внимание пользовательскому интерфейсу, чтобы сделать его удобным и интуитивно понятным. Также необходимо было сделать его внешне привлекательным, чтобы использование приложение вызывало у пользователя положительные эмоции.

Использование языка программирования C# и технологии Windows Forms обеспечит надежность и производительность приложения. Кроме того, разработка системы планирования "Ежедневник" позволит развивать навыки программирования и создания пользовательских приложений для операционной системы Windows. Создание дизайна приложение развивает творческие способности и насмотренность.

**Список использованной литературы**

1. Документация Microsoft по C#: <https://docs.microsoft.com/en-us/dotnet/csharp/>
2. Справочник по Windows Forms: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/>
3. Статьи на сайте CodeProject о разработке приложений на C# в Windows Forms: <https://www.codeproject.com/>
4. Лекции и уроки по C# на сайте MSDN: <https://docs.microsoft.com/en-us/learn/>
5. Книга "Programming C# 7.0: Building Windows 10 Applications": <https://www.amazon.com/Programming-C-7-0-Building-Applications/dp/1491987650>

**Приложение №1.**

1. Class Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Log\_in());

}

}

}

1. Class Users.cs

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Drawing;

using System.Drawing.Printing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WindowsFormsApp1

{

/// <summary>

/// Cведения о всех зарегистрированных пользователях

/// </summary>

public class Users

{

private int id;

private string login\_user;

private string password\_user;

public Users(string login\_user, string password\_user)

{

this.login\_user = login\_user;

this.password\_user = password\_user;

}

public int Id { get => id; set => id = value; }

public string Login\_user { get => login\_user; set => login\_user = value; }

public string Password\_user { get => password\_user; set => password\_user = value; }

public override string ToString()

{

return login\_user + " " + password\_user;

}

/// <summary>

/// Проверяем, есть ли пользователь в базе данных

/// </summary>

/// <returns></returns>

public bool CheckingUser()

{

using (SQLiteConnection connection = new SQLiteConnection("Data Source = diary.db"))

{

connection.Open();

string sql = $"SELECT \* FROM Users WHERE login\_user = '{this.login\_user}' AND password\_user = '{this.password\_user}'";

using (SQLiteCommand command = new SQLiteCommand(sql, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

this.id = Convert.ToInt32(reader["id"]);

return true;

}

else

{

return false;

}

}

}

}

}

/// <summary>

/// Добавляем нового пользователя

/// </summary>

public void UpdateUser()

{

using (SQLiteConnection connection = new SQLiteConnection("Data Source=diary.db"))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

command.CommandText = $"INSERT INTO Users (login\_user, password\_user) VALUES ('{this.login\_user}', '{this.password\_user}')";

command.ExecuteNonQuery();

}

connection.Close();

}

}

/// <summary>

/// Проверяет, зарегистрирован ли пользователь с таким именем (для регистрации)

/// </summary>

/// <returns></returns>

public bool ExistenceUser()

{

using (SQLiteConnection connection = new SQLiteConnection("Data Source=diary.db"))

{

connection.Open();

string query = $"SELECT COUNT(\*) FROM Users WHERE login\_user = '{this.login\_user}'";

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

int count = Convert.ToInt32(command.ExecuteScalar());

if (count > 0)

{

return true;

}

else

{

return false;

}

}

}

}

public Users\_Task userTask { get; set; }

}

1. }Class Users\_Task.cs

using System;

using System.Collections.Generic;

using System.Data.SQLite;

namespace WindowsFormsApp1

{

/// <summary>

/// Cведения о задачах всех зарегистрированных пользователях

/// </summary>

public class Users\_Task

{

private int task\_id;

private string task;

private DateTime deadline;

private int priority;

private int user\_id;

public Users\_Task(int task\_id, string task, DateTime deadline, int priority, int user\_id)

{

this.task\_id = task\_id;

this.task = task;

this.deadline = deadline;

this.priority = priority;

this.user\_id = user\_id;

}

public int Task\_Id { get => task\_id; set => task\_id = value; }

public string Task { get => task; set => task = value; }

public DateTime Deadline { get => deadline; set => deadline = value; }

public int Priority { get => priority; set => priority = value; }

public int User\_Id { get => user\_id; set => user\_id = value; }

/// <summary>

/// Чтение данных из таблицы Users\_Task в List<Users\_Task> tasks

/// </summary>

/// <param name="user\_id"></param>

/// <returns></returns>

public static List<Users\_Task> ReadFromDB(int user\_id)

{

List<Users\_Task> tasks = new List<Users\_Task>();

string connectionString = "Data Source=diary.db;";

string query = "SELECT task\_id, task, deadline, priority FROM Users\_Task WHERE user\_id = @user\_id";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@user\_id", user\_id);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

int taskId = reader.GetInt32(0);

string task = reader.GetString(1);

string deadlineText = reader["deadline"].ToString();

DateTime deadline = DateTime.Parse(deadlineText);

int priority = reader.GetInt32(3);

Users\_Task newTask = new Users\_Task(taskId, task, deadline, priority, user\_id);

tasks.Add(newTask);

}

}

}

}

return tasks;

}

}

}

1. Class Users\_Task.cs (Модульные тесты)

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Users\_TaskMS

{

public class Users\_Task

{

private int task\_id;

private string task;

private DateTime deadline;

private int priority;

private int user\_id;

public Users\_Task(int task\_id, string task, DateTime deadline, int priority, int user\_id)

{

this.task\_id = task\_id;

this.task = task;

this.deadline = deadline;

this.priority = priority;

this.user\_id = user\_id;

}

public int Task\_Id { get => task\_id; set => task\_id = value; }

public string Task { get => task; set => task = value; }

public DateTime Deadline { get => deadline; set => deadline = value; }

public int Priority { get => priority; set => priority = value; }

public int User\_Id { get => user\_id; set => user\_id = value; }

//Чтение данных из таблицы Users\_Task в List<Users\_Task> tasks

public static List<Users\_Task> ReadFromDB(int user\_id)

{

List<Users\_Task> tasks = new List<Users\_Task>();

string connectionString = "Data Source=TestDB.db;";

string query = "SELECT task\_id, task, deadline, priority FROM TestDB WHERE user\_id = @user\_id";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

command.Parameters.AddWithValue("@user\_id", user\_id);

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

int taskId = reader.GetInt32(0);

string task = reader.GetString(1);

string deadlineText = reader["deadline"].ToString();

DateTime deadline = DateTime.Parse(deadlineText);

int priority = reader.GetInt32(3);

Users\_Task newTask = new Users\_Task(taskId, task, deadline, priority, user\_id);

tasks.Add(newTask);

}

}

}

}

return tasks;

}

public static void Main(string[] args)

{

Users\_Task task1 = new Users\_Task(12, "Task1", new DateTime(2023, 12, 12), 4, 1);

}

}

}

1. Class UsersTests.cs (Модульные тесты)

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using Users\_TaskMS;

namespace UsersTests

{

[TestClass]

public class UsersTests

{

[TestMethod]

public void ReadFromDB\_ReturnsTasksList()

{

int testUserId = 1;

List<Users\_Task> expectedTasks = new List<Users\_Task>()

{

new Users\_Task(1, "Task 1", new DateTime(2022, 12, 31),1,1),

new Users\_Task(2, "Task 2", new DateTime(2022, 12, 31),2,1) ,

new Users\_Task(3, "Task 3", new DateTime(2022, 12, 31),3,1)

};

List<Users\_Task> actualTasks = Users\_Task.ReadFromDB(testUserId);

Assert.AreEqual(expectedTasks.Count, actualTasks.Count);

for (int i = 0; i < expectedTasks.Count; i++)

{

Assert.AreEqual(expectedTasks[i].Task\_Id, actualTasks[i].Task\_Id);

Assert.AreEqual(expectedTasks[i].Task, actualTasks[i].Task);

Assert.AreEqual(expectedTasks[i].Deadline, actualTasks[i].Deadline);

Assert.AreEqual(expectedTasks[i].Priority, actualTasks[i].Priority);

Assert.AreEqual(expectedTasks[i].User\_Id, actualTasks[i].User\_Id);

}

}

}

}

**Приложение №2.**

1. Код формы Sign\_up.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

/// <summary>

/// Регистрация новых пользователей

/// </summary>

public partial class Sign\_up : Form

{

/// <summary>

/// Инициализация формы Sign\_Up

/// </summary>

public Sign\_up()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// Устанавливает основные начальные параметры формы, такие как символ для скрытия пароля и видимость иконок

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Form2\_Load(object sender, EventArgs e)

{

textBox\_Password.PasswordChar = '\*';

pictureBox1.Visible = false;

textBox\_Login.MaxLength = 50;

textBox\_Password.MaxLength = 50;

}

/// <summary>

/// Показывает пароль при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox1\_Click(object sender, EventArgs e)

{

textBox\_Password.UseSystemPasswordChar = false;

pictureBox1.Visible = false;

pictureBox2.Visible = true;

}

/// <summary>

/// Скрывает пароль при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox2\_Click(object sender, EventArgs e)

{

textBox\_Password.UseSystemPasswordChar = true;

pictureBox1.Visible = true;

pictureBox2.Visible = false;

}

/// <summary>

/// Очищает все поля ввода при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

textBox\_Login.Text = "";

textBox\_Password.Text = "";

}

/// <summary>

/// Регистрирует нового пользователя при нажатии на кнопку "Зарегистрироваться"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (textBox\_Login.Text == "" || textBox\_Password.Text == "")

{

MessageBox.Show("Введиите данные");

return;

}

Users user = new Users(textBox\_Login.Text, textBox\_Password.Text);

if (user.ExistenceUser())

{

MessageBox.Show("Такой пользователь уже есть");

Log\_in b = new Log\_in();

b.Show();

this.Hide();

return;

}

else

{

user.UpdateUser();

MessageBox.Show("Аккаунт успешно создан!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

Log\_in a = new Log\_in();

a.Show();

this.Hide();

}

}

/// <summary>

/// Переход на форму авторизации при закрытии формы регистрации

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

}

}

1. Код формы Log\_in.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Diagnostics;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WindowsFormsApp1

{

/// <summary>

/// Обеспечивает вход пользователя в систему

/// </summary>

public partial class Log\_in : Form

{

/// <summary>

/// Инициализация формы

/// </summary>

public Log\_in()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// Устанавливает основные начальные параметры формы, такие как символ для скрытия пароля и видимость иконок

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Login\_in\_Load(object sender, EventArgs e)

{

textBox\_Password.PasswordChar = '\*';

pictureBox1.Visible = false;

textBox\_Login.MaxLength = 50;

textBox\_Password.MaxLength = 50;

}

/// <summary>

/// Входит в систему и переходит на форму Plan при нажатии на кнопку "Войти"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Users user = new Users(textBox\_Login.Text, textBox\_Password.Text);

if (user.CheckingUser())

{

this.Hide();

Plan m = new Plan(user.Id);

m.Show();

}

else

{

MessageBox.Show("Неверный логин или пароль!");

}

}

/// <summary>

/// Переходит на форму Sign\_up при нажатии на текст-ссылку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Sign\_up frm2 = new Sign\_up();

frm2.ShowDialog();

this.Hide();

}

/// <summary>

/// Скрывает пароль при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox2\_Click(object sender, EventArgs e)

{

textBox\_Password.UseSystemPasswordChar = true;

pictureBox1.Visible = true;

pictureBox2.Visible = false;

}

/// <summary>

/// Показывает пароль при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox1\_Click(object sender, EventArgs e)

{

textBox\_Password.UseSystemPasswordChar = false;

pictureBox1.Visible = false;

pictureBox2.Visible = true;

}

/// <summary>

/// Очищает все поля ввода при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

textBox\_Login.Text = "";

textBox\_Password.Text = "";

}

/// <summary>

/// Закрывает приложение полностью при закртии формы

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Log\_in\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

}

}

1. Код формы Plan.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SQLite;

using System.Data.SqlClient;

using System.Timers;

using System.Runtime.InteropServices;

using System.Reflection;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.TrackBar;

using System.Data.Entity;

using System.Runtime.CompilerServices;

using System.Windows.Forms.DataVisualization.Charting;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WindowsFormsApp1

{

/// <summary>

/// Выполнение основных функций "Ежедневника" таких, как отображение списка задач, поиск задач и др.

/// </summary>

public partial class Plan : Form

{

private int id;

int selectedRow;

/// <summary>

/// Инициализация формы Plan

/// </summary>

/// <param name="id"></param>

public Plan(int id)

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

this.id = id;

}

/// <summary>

/// Создание колонок DataGridView

/// </summary>

private void CreateColumns()

{

dataGridView1.Columns.Add("task\_id", "Номер задачи");

dataGridView1.Columns.Add("task", "Задача");

dataGridView1.Columns.Add("deadline", "Срок выполнения");

dataGridView1.Columns.Add("priority", "Приоритет");

dataGridView1.Columns.Add("userid", "Пользователь");

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[4].Visible = false;

}

/// <summary>

/// Обновление DataGridView (очистка полей и новая запись из List)

/// </summary>

/// <param name="dgw"></param>

private void RefreshDataGridView(DataGridView dgw)

{

dgw.Rows.Clear();

List <Users\_Task> Tasks = Users\_Task.ReadFromDB(id);

foreach (Users\_Task usertask in Tasks)

{

dgw.Rows.Add(usertask.Task\_Id, usertask.Task, usertask.Deadline, usertask.Priority, usertask.User\_Id);

}

}

/// <summary>

/// Поиск в базе данных по введёному в поле тексту

/// </summary>

/// <param name="dgw"></param>

private void Search(DataGridView dgw)

{

dgw.Rows.Clear();

var connection = new SQLiteConnection("Data Source = diary.db");

string searchstring = $"select \* from Users\_Task where (task like '%" + textBox\_Search.Text + "%' OR deadline like '%" + textBox\_Search.Text + "%' OR priority like '%" + textBox\_Search.Text + $"%' ) AND user\_id = '{id}'";

connection.Open();

var command = connection.CreateCommand();

command.CommandText = searchstring;

var reader = command.ExecuteReader();

while (reader.Read())

{

int taskId = reader.GetInt32(0);

string task = reader.GetString(1);

string deadlineText = reader["deadline"].ToString();

DateTime deadline = DateTime.Parse(deadlineText);

int priority = reader.GetInt32(3);

dgw.Rows.Add(taskId, task, deadline, priority, id);

}

connection.Close();

}

/// <summary>

/// Поиск задачи при введение текста в TextBox

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void textBox\_Search\_TextChanged(object sender, EventArgs e)

{

Search(dataGridView1);

}

/// <summary>

/// Очистка всех полей для ввода

/// </summary>

private void ClearTextBox()

{

textBox1\_Task\_id.Text = "";

textBox1\_task.Text = "";

textBox3\_time.Text = "";

textBox2\_priority.Text = "";

textBox\_Search.Text = "";

}

/// <summary>

/// Очистка всех полей при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox4\_Click(object sender, EventArgs e)

{

ClearTextBox();

}

/// <summary>

/// Удаление выбранной строки из DataGridView и базы данных

/// </summary>

/// <param name="user\_id"></param>

private void DeleteSelectedRow(int user\_id)

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления! \n" +

"Выберите для этого крайнюю левую ячейку в строке нужной задачи!", "Укажите строку!", MessageBoxButtons.OK, MessageBoxIcon.Question);

return;

}

int selectedTaskId = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["task\_id"].Value);

string connectionString = "Data Source=diary.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

string deletingstring = $"DELETE FROM Users\_Task WHERE user\_id= '{id}' AND task\_id='{selectedTaskId}'";

command.CommandText = deletingstring;

command.ExecuteNonQuery();

}

connection.Close();

}

dataGridView1.Rows.Remove(dataGridView1.SelectedRows[0]);

MessageBox.Show("Задача успешно удалена!");

}

/// <summary>

/// Удаляет задачу из базы данных и DataGridView при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_delete\_Click(object sender, EventArgs e)

{

DeleteSelectedRow(id);

ClearTextBox();

RefreshDataGridView(dataGridView1);

}

/// <summary>

/// Обновляет данные в DataGridView и в базе данных

/// </summary>

private void ChangeData()

{

int task\_id;

if (!int.TryParse(textBox1\_Task\_id.Text, out task\_id))

{

MessageBox.Show("Введите корректное значение для Task ID", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string task = textBox1\_task.Text;

DateTime deadline;

if (!DateTime.TryParse(textBox3\_time.Text, out deadline))

{

MessageBox.Show("Введите корректное значение для срока исполнения", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int priority;

if (!int.TryParse(textBox2\_priority.Text, out priority))

{

MessageBox.Show("Введите корректное значение для Priority", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else

{

if (!(0 <= priority && priority <= 10))

{

MessageBox.Show("Введите значение приоритета в пределах от 0 до 10!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

foreach (DataGridViewRow row in dataGridView1.SelectedRows)

{

row.Cells["task\_id"].Value = task\_id;

row.Cells["task"].Value = task;

row.Cells["deadline"].Value = deadline;

row.Cells["priority"].Value = priority;

}

string connectionString = "Data Source=diary.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

string updatingquery = $"UPDATE Users\_Task SET task = '{task}', deadline = '{deadline}', priority = '{priority}' WHERE task\_id = '{task\_id}' and user\_id = '{id}'";

command.CommandText = updatingquery;

command.ExecuteNonQuery();

}

connection.Close();

}

MessageBox.Show("Данные успешно обновлены", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

RefreshDataGridView(dataGridView1);

}

/// <summary>

/// Изменяет данные задачи в базе данных и DataGridView

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_change\_Click(object sender, EventArgs e)

{

ChangeData();

ClearTextBox();

}

/// <summary>

/// Получение загруженности недели

/// </summary>

/// <param name="tasks"></param>

private void GetCountTasks(List<Users\_Task> tasks)

{

Dictionary<string, int> tasksPerDay = new Dictionary<string, int>();

DateTime now = DateTime.Now.Date;

DateTime endDate = now.AddDays(7).Date;

for (DateTime date = now; date < endDate; date = date.AddDays(1))

{

int count = tasks.Count(t => t.Deadline.Date == date.Date);

tasksPerDay.Add(date.ToShortDateString(), count);

}

foreach (KeyValuePair<string, int> pair in tasksPerDay)

{

chart1.Series[0].Points.AddXY(pair.Key, pair.Value);

}

string maxTasksDay = string.Join(", ", tasksPerDay.Where(x => x.Value == tasksPerDay.Max(y => y.Value)).Select(z => z.Key));

textBox\_Day.Text = maxTasksDay;

int maxTasks = tasksPerDay.Max(x => x.Value);

textBox\_CountDay.Text = Convert.ToString(maxTasks);

int misstask = tasks.Count(t => t.Deadline < DateTime.Now);

textBox\_misstask.Text = Convert.ToString(misstask);

}

/// <summary>

/// Получение загруженности недели при нажатии на кнопку "Узнать загруженность"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button1\_Count\_Tasks\_Click(object sender, EventArgs e)

{

List<Users\_Task> Tasks = Users\_Task.ReadFromDB(id);

chart1.Series[0].Points.Clear();

GetCountTasks(Tasks);

}

/// <summary>

/// Инициализация DataGridView

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void DataGrid\_Form\_Load(object sender, EventArgs e)

{

CreateColumns();

RefreshDataGridView(dataGridView1);

Search(dataGridView1);

}

/// <summary>

/// Вывод данных о задаче в поля, расположенный снизу формы Plan

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridViewRow row = dataGridView1.Rows[selectedRow];

textBox1\_Task\_id.Text = row.Cells[0].Value.ToString();

textBox1\_task.Text = row.Cells[1].Value.ToString();

textBox3\_time.Text = row.Cells[2].Value.ToString();

textBox2\_priority.Text = row.Cells[3].Value.ToString();

}

}

/// <summary>

/// Обновление таблицы DataGridView

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox3\_Click(object sender, EventArgs e)

{

RefreshDataGridView(dataGridView1);

ClearTextBox();

}

/// <summary>

/// Определение новой задачи, переход на форму Newtask

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_add\_Click(object sender, EventArgs e)

{

Newtask = new Newtask(id);

newtask.Show();

}

/// <summary>

/// Полное закрытие приложения "Ежедневник"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Plan\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

}

}

1. Код формы Newtask.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Data.SQLite;

using System.Globalization;

namespace WindowsFormsApp1

{

/// <summary>

/// Создание новой задачи, добавление её в базу данных

/// </summary>

public partial class Newtask : Form

{

private int id;

/// <summary>

/// Инициализация формы Newtask, определение её положение на экране

/// </summary>

/// <param name="id"></param>

public Newtask(int id)

{

this.id = id;

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// Добавление в базу данных новой задачи при заполнение полей ввода и нажатии на кнопку "Сохранить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void button\_save\_Click(object sender, EventArgs e)

{

var task = textBox1\_task.Text;

var priority = textBox2\_priority.Text;

var data = textBox3\_time.Text;

DateTime time;

SQLiteConnection connection = new SQLiteConnection("Data Source = diary.db");

connection.Open();

if (DateTime.TryParse(data, out time))

{

if (0<=Convert.ToInt32(priority) && Convert.ToInt32(priority) <= 10)

{

var newtaskquery = $"insert into Users\_Task(task,deadline, priority, user\_id) values('{task}','{time}','{priority}', '{id}')";

var command = connection.CreateCommand();

command.CommandText = newtaskquery;

command.ExecuteNonQuery();

MessageBox.Show("Запись успешно создана!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Неверное значение приоритета! \n Укажите значение от 0 до 10!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

else

{

MessageBox.Show("Неверно введено время исполнения!","Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

connection.Close();

}

/// <summary>

/// Очистка полей ввода при нажатии на кнопку

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void pictureBox4\_Click(object sender, EventArgs e)

{

textBox1\_task.Text = "";

textBox2\_priority.Text = "";

textBox3\_time.Text = "";

}

}

}