Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования «Уральский федеральный университет   
имени первого Президента России Б. Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

**Отчёт по Лабораторной работе №4**

по дисциплине «Средства управления информационными ресурсами АС»

**Тема:** Представление данных связанных отношением «многие ко многим»

Научный руководитель Парфенов Ю. П.

Курс, группа РИЗ-300016у

Студент Кулаков М. И.

Екатеринбург, 2022

# СОДЕРЖАНИЕ

1 Постановка задачи 3

2 Образы экранных форм в конструкторе и исполняемой программе с пояснениями элементов управления 3

3 Добавленный программный код с пояснениями 4

4 Подробное описание впервые использованных свойств и методов 7

# 1 Постановка задачи

На четвертой вкладке «Лабораторная работа №4» постройте диалог, демонстрирующий работу со строками, связанными отношением «многие ко многим». Для этого источниками данных для двух объектов ***DataGridView*** являются справочная и главная таблицы индивидуального задания.

1. Напишите программный код, создающий необходимые объекты и задающий их свойства.
2. Приведите подробный комментарий для каждого использованного оператора программы.
3. Проверьте работу по управлению данными в полученной форме.

# 2 Образы экранных форм в конструкторе и исполняемой программе с пояснениями элементов управления

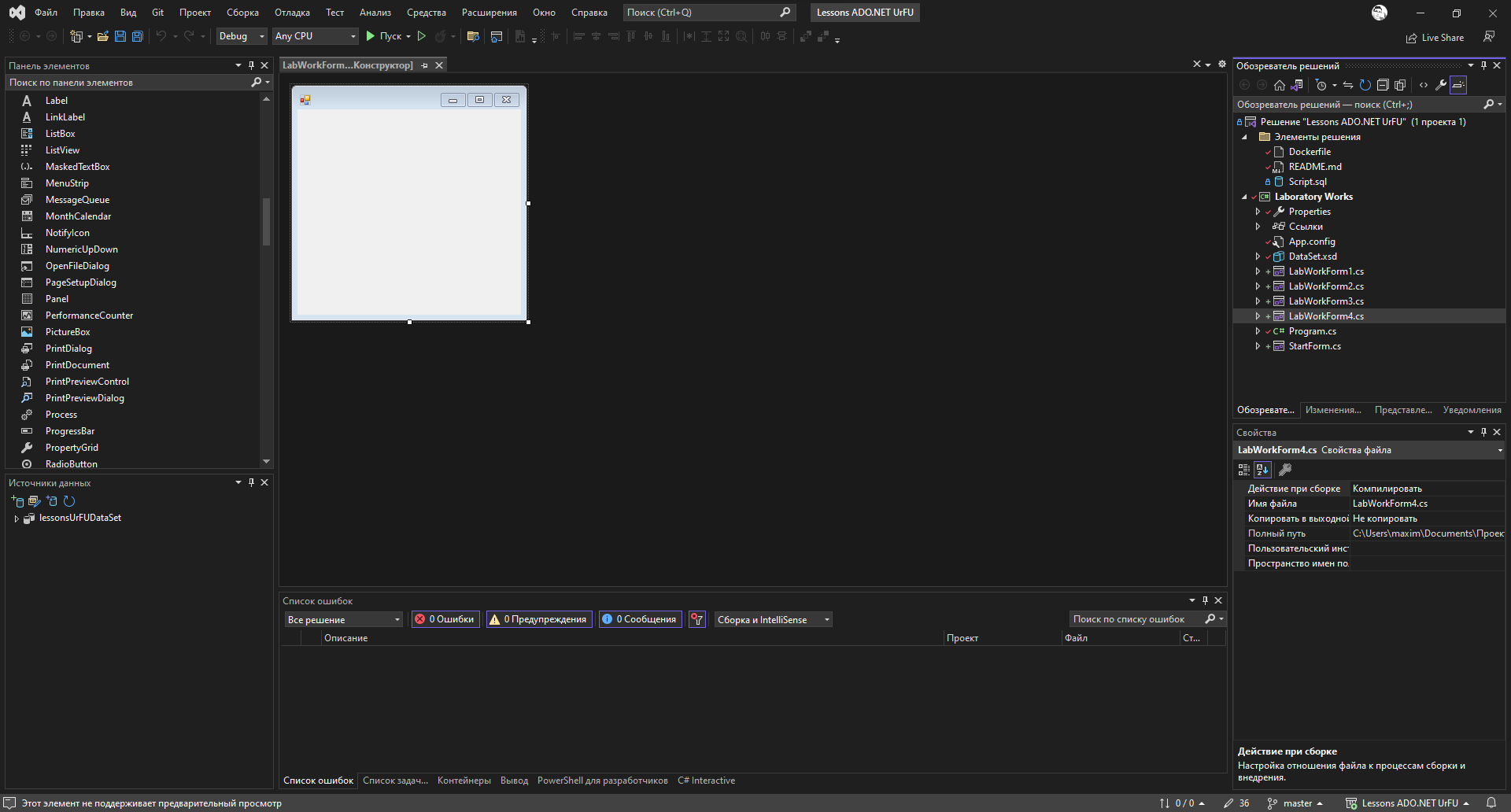


Рисунок №1 – Экранная форма в конструкторе

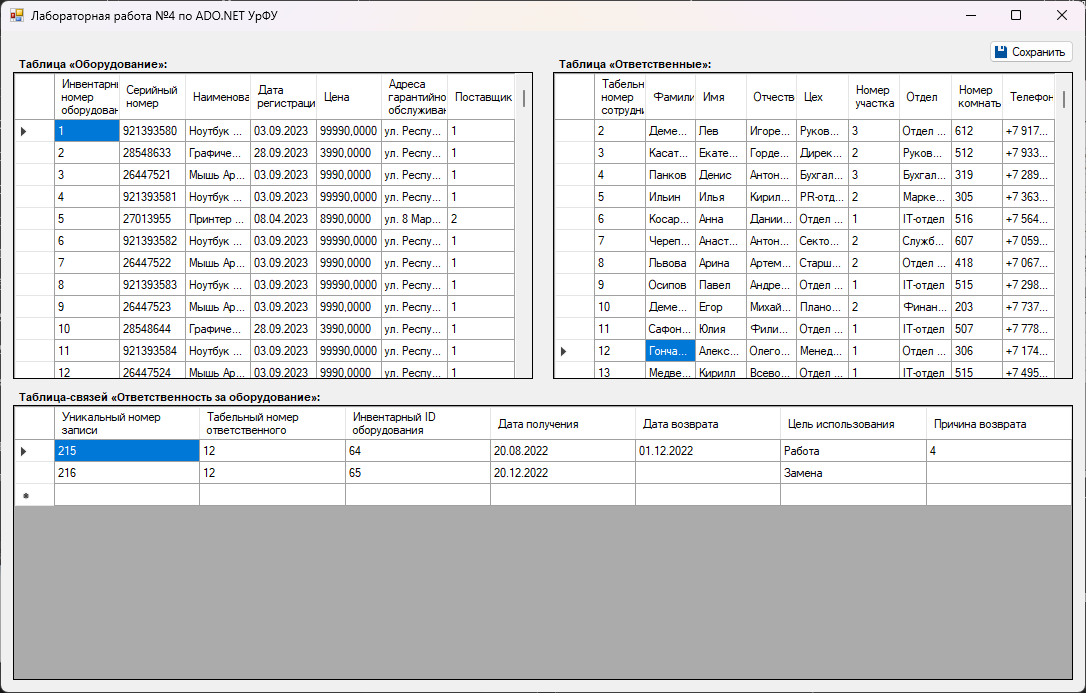


Рисунок №2 – Исполняемая программа

Согласно заданию, с помощью конструктора форм не были добавлены какие-либо объекты. Все объекты были добавлены на форму с помощью кода. На форме находятся:

* три текстовые метки, класса ***Label***;
* три табличных представления данных, класса ***DataGridView***;
* одна кнопка для отправки изменений на SQL‑сервер, класса ***Button***.

**\_dataGridViewResponsibility** теперь используется в режиме ReadOnly.

# 3 Добавленный программный код с пояснениями

По сравнению с прошлой лабораторной работой, был изменён конструктор формы. Из него была удалён код, отвечающий за получение данных с таблицы «Ответственность за оборудование». Получение данных для этой таблицы выполняется в рамках метода **GetResponsibilityData(object, EventArgs)**. Данный метод используется в качестве обработчика события **SelectionChanged** для табличных представлений **\_dataGridViewEquipment** и **\_dataGridViewResponsible**. При формировании SQL-запроса используется свойство **Parameters** класса ***SqlCommand***.

Код конструктора формы:

public LabWorkForm4()

{

InitializeComponent();

// Настроки формы

this.Text = "Лабораторная работа №4 по ADO.NET УрФУ";

this.MinimumSize = new Size(1100, 700);

this.MaximumSize = new Size(1100, 700);

this.FormBorderStyle = FormBorderStyle.FixedSingle;

this.StartPosition = FormStartPosition.CenterScreen;

// Добавление текстовых меток на форму

this.Controls.Add(\_labelEquipment);

this.Controls.Add(\_labelResponsible);

this.Controls.Add(\_labelResponsibility);

// Добавление DataGridView-ов на форму

this.Controls.Add(\_dataGridViewEquipment);

this.Controls.Add(\_dataGridViewResponsible);

this.Controls.Add(\_dataGridViewResponsibility);

// Добавление кнопки на форму

this.Controls.Add(\_buttonSave);

// --------------------------------

// Работа с базой данных ----------

// --------------------------------

// Создание запросов на получение данных

var commands = new[]

{

new SqlCommand(@"SELECT InventoryID AS N'Инвентарный номер оборудования',

SerialNumber AS N'Серийный номер',

EquipmentName AS N'Наименование',

RegistrationDate AS N'Дата регистрации',

Price AS N'Цена',

WarrantyServiceAddress AS N'Адреса гарантийного обслуживания',

Supplier AS N'Поставщик'

FROM Equipment;", \_sqlConnection),

new SqlCommand(@"SELECT PersonnelNumber AS N'Табельный номер сотрудника',

Surname AS N'Фамилия',

Name AS N'Имя',

MiddleName AS N'Отчество',

Workshop AS N'Цех',

LotNumber AS N'Номер участка',

Department AS N'Отдел',

RoomNumber AS N'Номер комнаты',

Telephone AS N'Телефон'

FROM Responsible;", \_sqlConnection)

};

\_sqlConnection.Open();

try

{

// Создания SQL адаптеров на основе ранее созданных объектов-команд

\_dataAdapterEquipment = new SqlDataAdapter(commands[0]);

\_dataAdapterResponsible = new SqlDataAdapter(commands[1]);

var dataAdapters = new[]

{

\_dataAdapterEquipment,

\_dataAdapterResponsible

};

var dataTables = new[]

{

\_dataTableEquipment,

\_dataTableResponsible

};

var dataGridViews = new[]

{

\_dataGridViewEquipment,

\_dataGridViewResponsible

};

for (int i = 0; i < 2; i++)

{

dataTables[i].Reset();

// Отпрака запроса

dataAdapters[i].Fill(dataTables[i]);

// Связка DataGridView с DataTable

dataGridViews[i].DataSource = dataTables[i];

dataGridViews[i].SelectionChanged += GetResponsibilityData;

}

}

finally

{

\_sqlConnection.Close();

}

GetResponsibilityData(\_dataGridViewEquipment, null);

\_buttonSave.Click += SaveData;

}

}

Код метода для получения данных из таблицы «Ответственность за оборудование»:

/// <summary>

/// Получение данных с таблицы «Ответственность за оборудование»

/// </summary>

private void GetResponsibilityData(object sender, EventArgs e)

{

\_sqlConnection.Open();

try

{

// Создание запроса на получение данных

string tableJoin = (sender == \_dataGridViewEquipment)

? @"FROM Equipment LEFT OUTER JOIN Responsibility AS R ON R.Equipment = Equipment.InventoryID WHERE R.Equipment = @Equipment"

: @"FROM Responsible LEFT OUTER JOIN Responsibility AS R ON R.Responsible = Responsible.PersonnelNumber WHERE R.Responsible = @Responsible";

SqlCommand sqlCommandResponsibility = new(@"SELECT R.Id AS N'Уникальный номер записи', R.Responsible AS N'Табельный номер ответственного', R.Equipment AS N'Инвентарный ID оборудования', R.DateOfReceiving AS N'Дата получения', R.ReturnDate AS N'Дата возврата', R.PurposeOfUse AS N'Цель использования', R.ReasonForReturn AS N'Причина возврата'" + tableJoin, \_sqlConnection);

\_dataAdapterResponsibility = new SqlDataAdapter(sqlCommandResponsibility);

// Добавление параметра в запрос в зависимости от того, какой DataGridView имеет фокус

if (sender as DataGridView == \_dataGridViewEquipment)

{

if (\_dataGridViewEquipment.CurrentRow == null) return;

var equipmentId = \_dataGridViewEquipment.CurrentRow.Cells[0].Value;

sqlCommandResponsibility.Parameters.Add("@Equipment", SqlDbType.Int);

sqlCommandResponsibility.Parameters["@Equipment"].Value = equipmentId;

}

else

{

if (\_dataGridViewResponsible.CurrentRow == null) return;

var responsibleId = \_dataGridViewResponsible.CurrentRow.Cells[0].Value;

sqlCommandResponsibility.Parameters.Add("@Responsible", SqlDbType.Int);

sqlCommandResponsibility.Parameters["@Responsible"].Value = responsibleId;

}

\_dataTableResponsibility.Clear(); // Очистка табличного представления

\_dataAdapterResponsibility.Fill(\_dataTableResponsibility); // Отправка запроса

\_dataGridViewResponsibility.DataSource = \_dataTableResponsibility; // Связка DataGridView с DataTable

}

finally

{

\_sqlConnection.Close();

}

}

# 4 Подробное описание впервые использованных свойств и методов

Методы класса ***SqlCommand***:

* **CreateCommand()**: возвращает созданный объект класса ***SqlCommand*** с указанным текстом команды и подключением к БД.

Методы класса ***SqlDataAdapter***:

* **CreateParameter()**: возвращает созданный объект класса ***SqlParameter***.

Свойства класса ***SqlDataAdapter***:

* **Parameters**: возвращает коллекцию класса ***SqlParameterCollection***, содержащую параметры, используемые в запросе ***SqlCommand***.
* **CommandText**: возвращает или задаёт текст команды ***SqlCommand***.

Свойства класса ***DataGridView***:

* **CurrentRow**: возвращает объект класса ***DataGridViewRow***, представляющий текущую строку в ***DataGridView***.
* **ReadOnly**: возвращает или задаёт значение, указывающее доступен ли ***DataGridView*** только для чтения.