Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция
по выполнению лабораторной работы №34
«Интеграция отчетов в приложение»

Минск 2024 г.

Лабораторная работа № 34

Тема работы: «Интеграция отчетов в приложение»

1 Цель работы

Сформировать умения и навыки организации интеграции отчетов в приложении

2 Задание

Реализовать генерацию отчетов в приложении из лабораторной работы 30 на основе теоретических сведений.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

Интеграция отчетов в приложение на C# может быть осуществлена с использованием различных инструментов и библиотек.

Одним из наиболее популярных инструментов для создания отчетов в C# является библиотека Crystal Reports, но существуют и другие варианты, такие как Microsoft Report Viewer, Stimulsoft Reports, и др.

Для дальнейшего примера будем использовать itext7 – библиотека для генерации PDF.

Для того, чтобы использовать данный инструмент, его необходимо установить и добавить в проект.

Для установки можно использовать два варианта – через GUI (менеджер пакетов NuGet) или через консоль менеджера пакетов NuGet.

Для установки пакета будет использован первый способ. Для открытия окна по установке пакетов NuGet необходимо выбрать пункт меню Проект (Project), далее нажать Управление пакетами NuGet (Manage NuGet Packages) (рисунок 1).

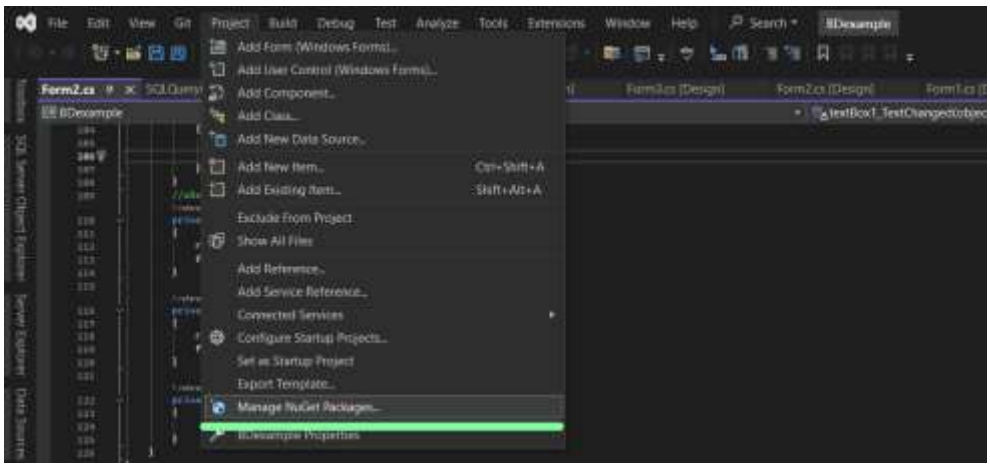

Рисунок 1 – Открытие консоли менеджера пакетов

Далее необходимо ввести в поисковую строку название пакета для установки – itext7 и нажать кнопку Установить (рисунок 2).

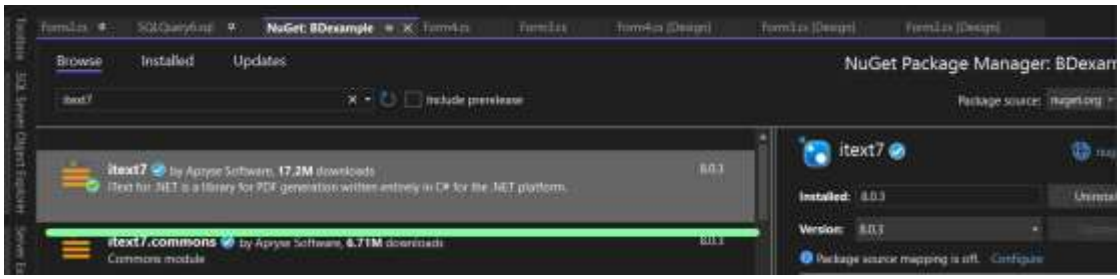

Рисунок 2 – Установка пакета itext7

После того, как пакет был успешно установлен, можно приступать к разработке отчета.

Для этого, на главной форме приложения создадим кнопку Отчет по заказам (рисунок 3).

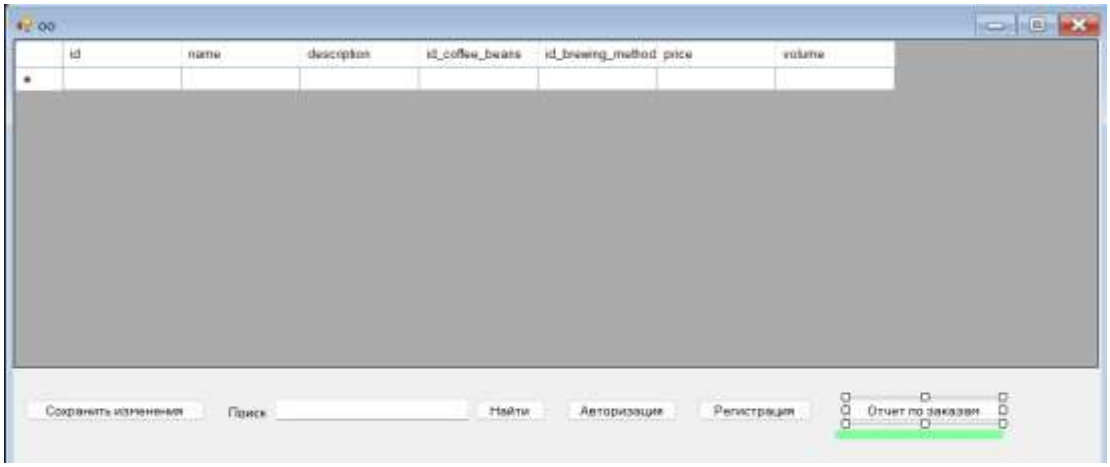

Рисунок 3 – Размещение кнопки «Отчет по заказам» на главной форме приложения

После этого добавим новую форму в проект и для обработчика события нажатия кнопки Отчет по заказам напишем код открытия новой формы:

```
Form5 form5 = new Form5();
form5.Show();
```

Далее, можем приступать к формированию отчета. Для этого в коде формы, отвечающей за генерацию отчета, добавим 3 строки для указания используемых модулей (рисунок 4).

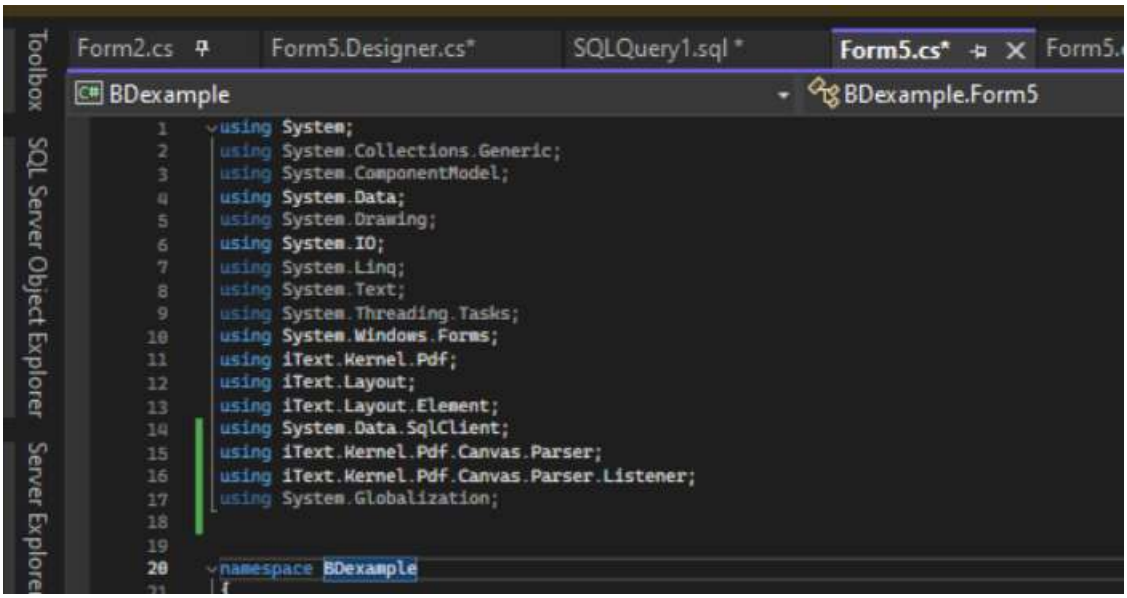

Рисунок 4 – Подключение пространств имен и классов к проекту

Кроме этого, чтобы избежать исключения «NotSupportedException: Either com.itextpdf:bouncy-castle-adapter or com.itextpdf:bouncy-castle-fips-adapter dependency must be added in order to use BouncyCastleFactoryCreator» необходимо через консоль управления пакетами NuGet установить Bouncy Castle Adapter (рисунки 5-6). Для этого нам необходима команда Install-Package itext7.bouncy-castle-adapter.

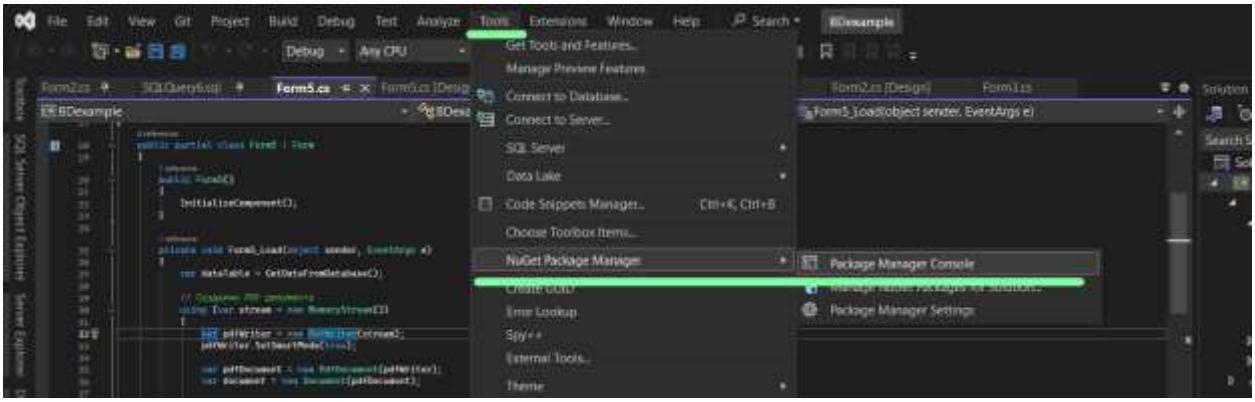

Рисунок 5 – Открытие Package Manager Console

Рисунок 6 – Установка Bouncy Castle Adapter

После добавления адаптера в проект в коде формы для генерации и просмотра отчета пропишем создание отчета по таблице Orders.

Для этого зарегистрируем событие открытия формы. Нажмем на форму и в окне событий для формы найдем событие Load (рисунок 7).

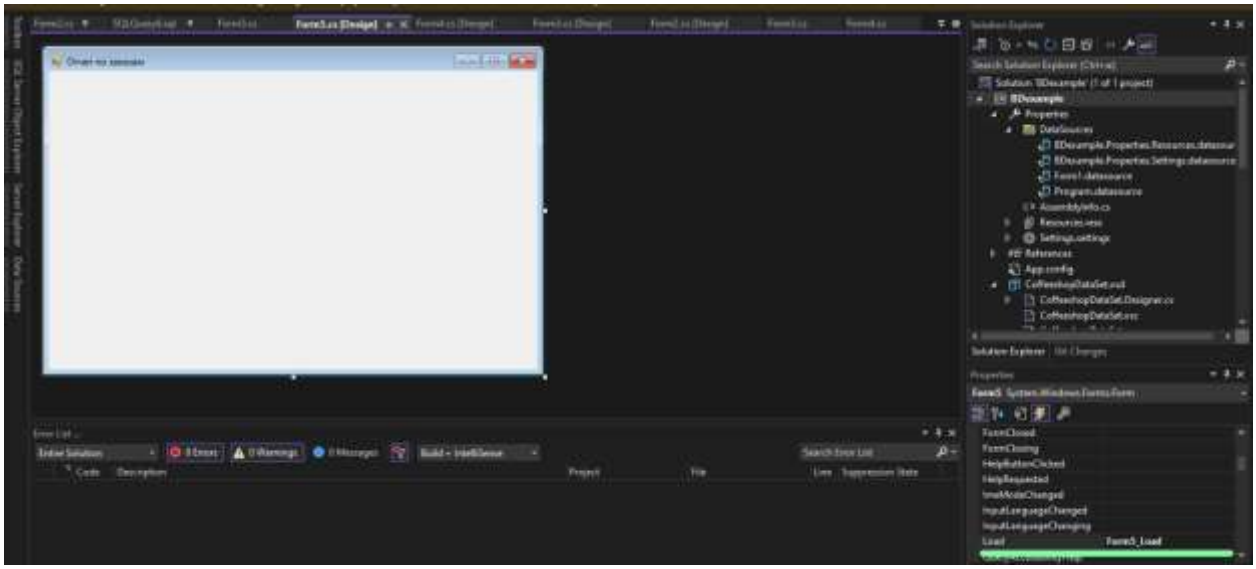

Рисунок 7 – Регистрация события Load для формы

Дважды кликаем по пустому полю рядом с названием события и попадаем в обработчик. В обработчике необходимо прописать следующий код:

```
var dataTable = GetDataFromDatabase();

// Создание PDF-документа
using (var stream = new MemoryStream())
{
    var pdfWriter = new PdfWriter(stream);
    pdfWriter.SetSmartMode(true);

    var pdfDocument = new PdfDocument(pdfWriter);
    var document = new Document(pdfDocument);
```

```
            // Заголовок
            document.Add(new Paragraph("Отчет о заказах"));

            // Таблица с данными
            var table = new Table(dataTable.Columns.Count);
            foreach (DataColumn column in dataTable.Columns)
            {
                table.AddHeaderCell(new Cell().Add(new Paragraph(column.ColumnName))); // формируем заголовки столбцов таблицы на основании название столбцов для таблицы Orders
            }

            foreach (DataRow row in dataTable.Rows)
            {
                foreach (object item in row.ItemArray)
                {
                    table.AddCell(new Cell().Add(new Paragraph(item.ToString()))); // заполняем строки таблицы в документе данными из таблицы Orders
                }
            }

            document.Add(table); // добавляем получившуюся таблицу в pdf-документ

            // Закрываем документ
            document.Close();

            // Сохраняем PDF-файл
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "PDF files (*.pdf)|*.pdf";
            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
                File.WriteAllBytes(saveFileDialog.FileName, stream.ToArray());
                MessageBox.Show("PDF-файл успешно создан!");
            }
        }
        DataTable GetDataFromDatabase() // получаем данные из таблицы Orders для pdf-документа
        {

            DataTable dataTable1 = new DataTable();

            using (SqlConnection connection = new SqlConnection("Data Source=(localdb)\\ProjectModels;Initial Catalog=Coffeeshop;Integrated Security=True"))
```

```
            {
                connection.Open();

                string sqlQuery = " SELECT id, id_drink, quantity, total FROM Orders where CONVERT(DATE, date_time, 101) = '2024-02-14 ";

                using (SqlDataAdapter adapter = new SqlDataAdapter(sqlQuery, connection))
                {
                    adapter.Fill(dataTable1);
                }
            }

            return dataTable1;
        }
    }
```

После этого запустим наш проект и проверим работоспособность создания отчета. Для начала, сохраним файл под именем Заказы (рисунок 8).

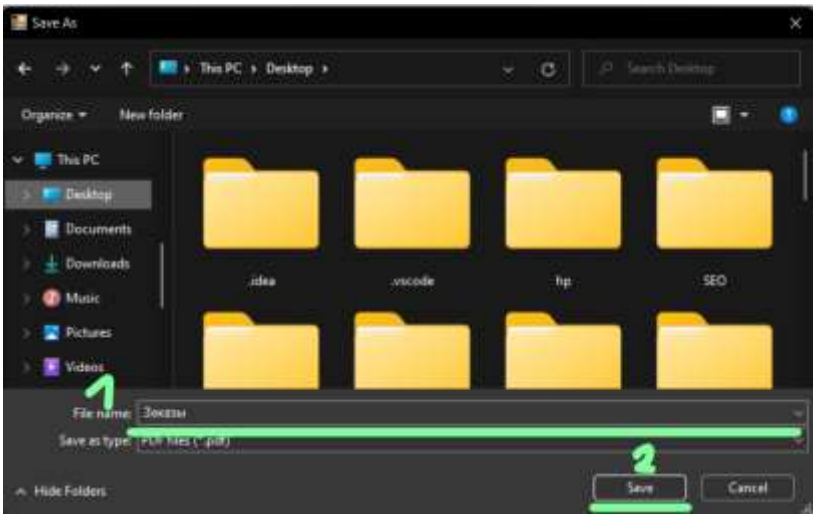

Рисунок 8 – Открытие диалогового окна для сохранения файла

Теперь осталось проверить, имеются ли в файле данные из таблицы Orders (рисунки 9-10).

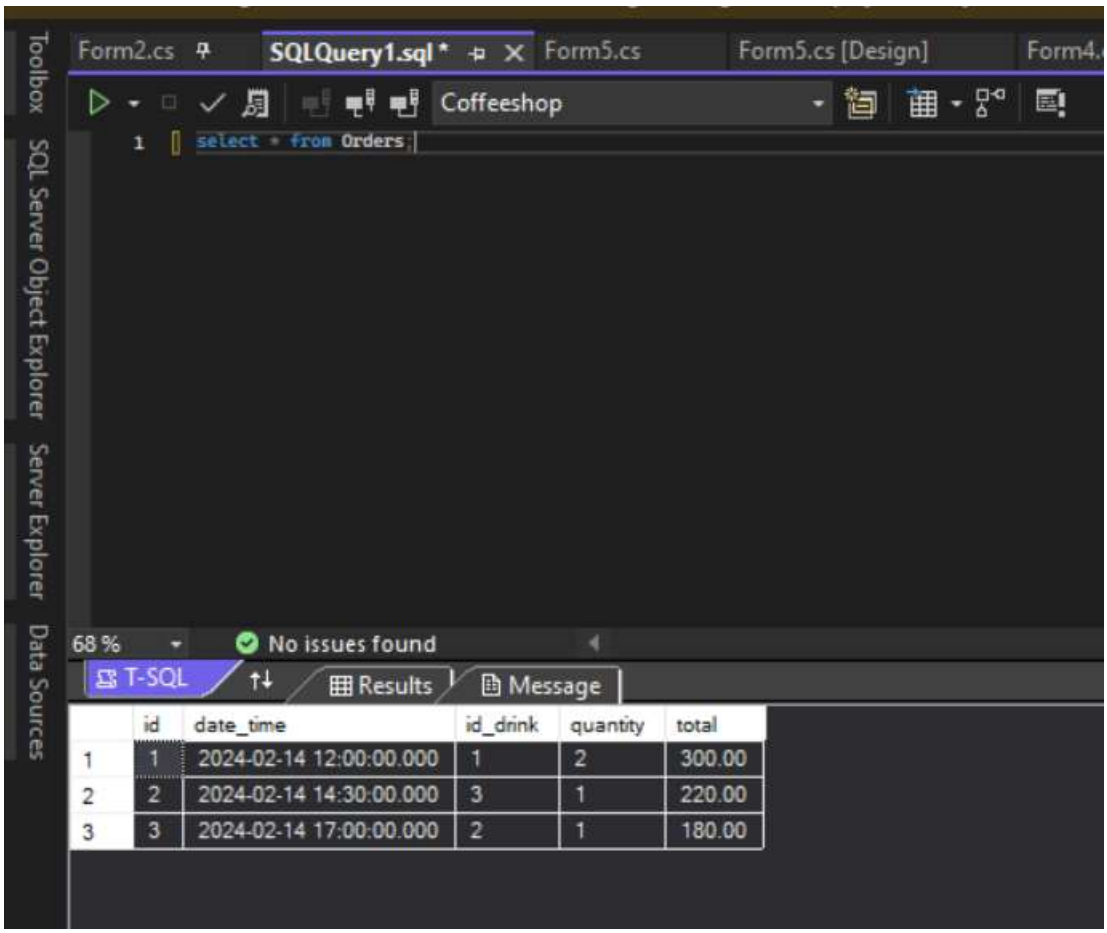

Рисунок 9 – Содержимое таблицы Orders

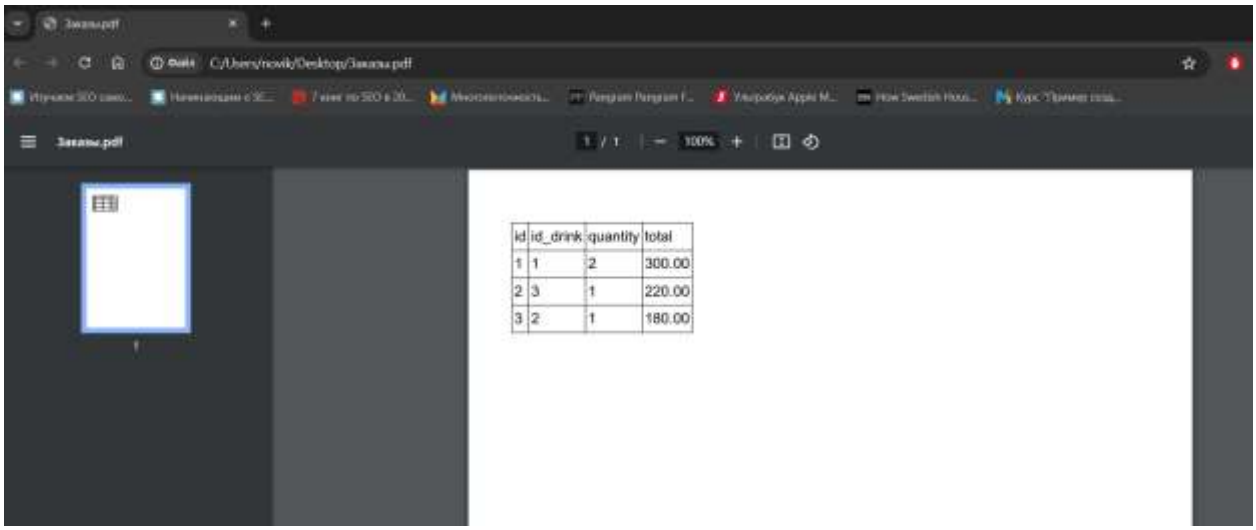

Рисунок 10 – Содержимое отчета на основе таблицы Orders

Как можно заметить, все необходимые нам данные из таблицы БД отображаются и в отчете.

Добавим две дополнительные функции – открытие сохраненного отчета в форме и вывод отчета на печать.

Для просмотра отчета на форме понадобятся два компонента – button и dataGridView из разделов Common Controls и Data панели Toolbox.

После этого форма должна быть оформлена в виде, представленном на рисунке 11.

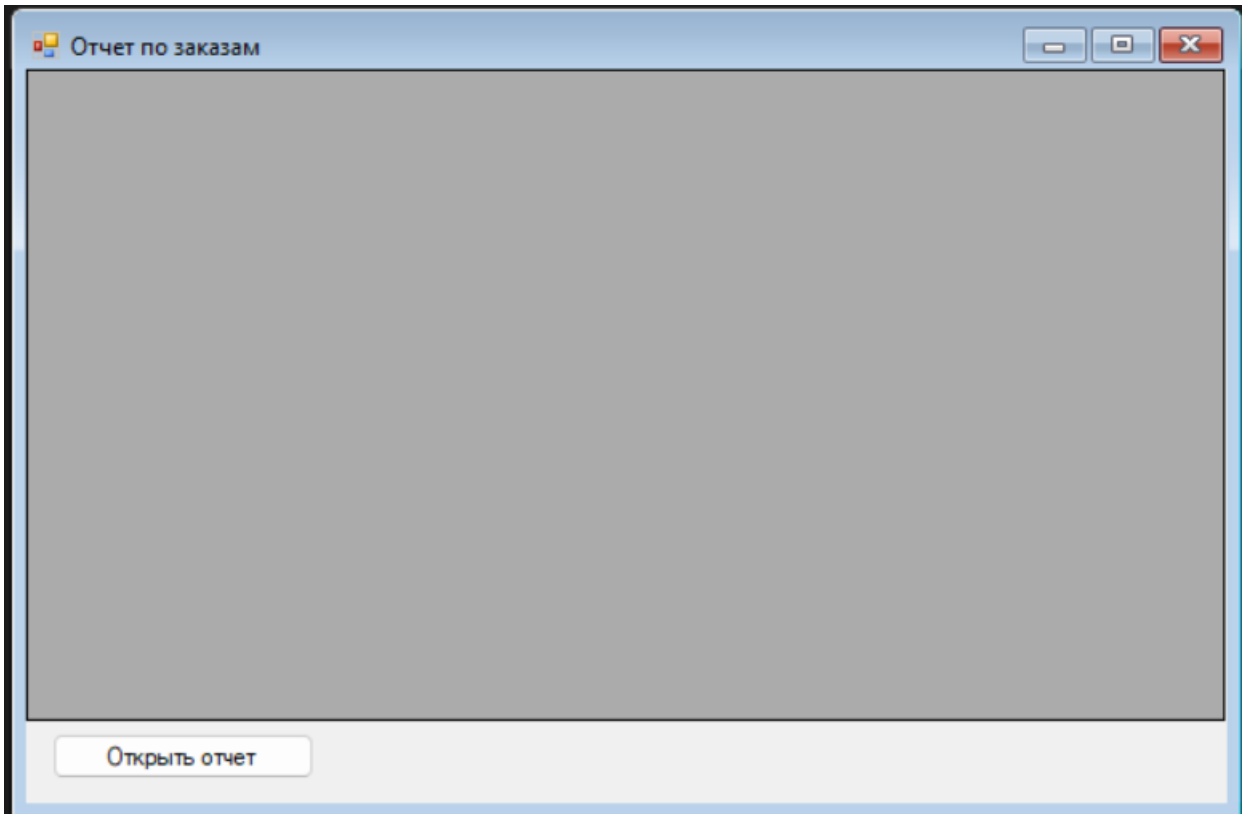

Рисунок 11 – Форма для просмотра отчетов

Для кнопки зарегистрируем событие на щелчок по ней и в обработчике напишем следующий код:

```
OpenFileDialog openFileDialog = new OpenFileDialog();
openFileDialog.Filter = "PDF files (*.pdf)|*.pdf";

if (openFileDialog.ShowDialog() == DialogResult.OK)
{
   string pdfFilePath = openFileDialog.FileName;
   DataTable dataTable = ExtractPdfData(pdfFilePath);

   dataGridView1.DataSource = dataTable;
}
```

В коде участвует пользовательский метод ExtractPdfData. Его код находится ниже:

```
private DataTable ExtractPdfData(string pdfFilePath)
{
   DataTable dataTable = new DataTable();
```

```
            // Add columns based on the table headers in the **second** row of the
PDF
            dataTable.Columns.Add("Номер");
            dataTable.Columns.Add("Номер напитка");
            dataTable.Columns.Add("Количество");
            dataTable.Columns.Add("Стоимость всего");

            using (PdfReader reader = new PdfReader(pdfFilePath))
            {
               PdfDocument pdfDoc = new PdfDocument(reader);

               for (int i = 1; i <= pdfDoc.GetNumberOfPages(); i++)
               {
                  var strategy = new SimpleTextExtractionStrategy();
                  string                                pageText                                =
PdfTextExtractor.GetTextFromPage(pdfDoc.GetPage(i), strategy);

                  string[] lines = pageText.Split('\n');

                  // Skip the first line (header)
                  for (int j = 1; j < lines.Length; j++)
                  {
                     string[] cells = lines[j].Split(' ');

                     DataRow row = dataTable.NewRow();
                     for (int k = 0; k < cells.Length; k++)
                     {
                        if (k < dataTable.Columns.Count)
                        {
                           row[k] = cells[k].Trim();
                        }
                        else
                        {
                           break;
                        }
                     }

                     dataTable.Rows.Add(row);
                  }
               }
            }

            return dataTable;
         }
```

Далее, реализуем функцию печати отчета. Для этого на форму положим еще один компонент button (рисунок 12).

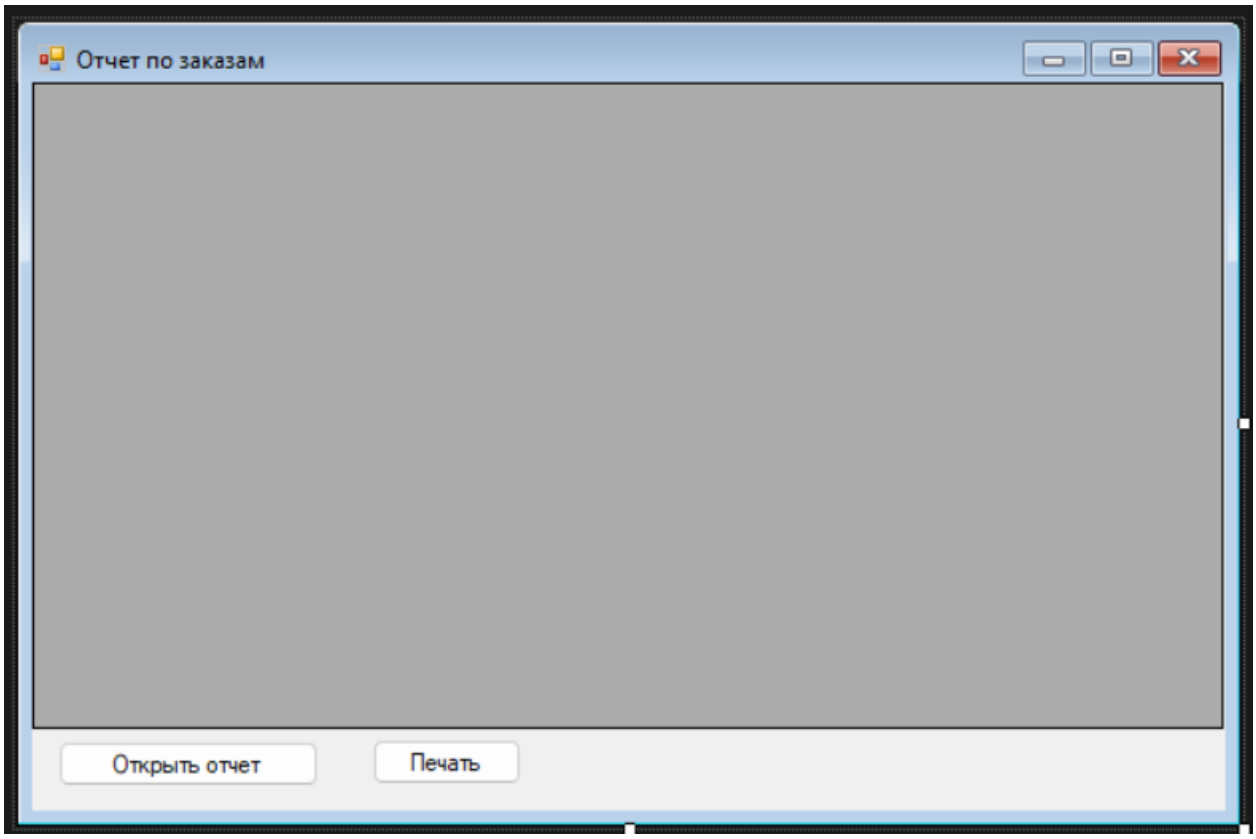

Рисунок 12 – Добавление кнопки Печать на форму

Зарегистрируем событие нажатия кнопки и напишем реализацию функции печати. Для этого будем использовать диалоговые окна открытия файла и печати.

Код обработчика события кнопки представлен ниже:

```
// Open PDF file
OpenFileDialog openFileDialog = new OpenFileDialog();
openFileDialog.Filter = "PDF files (*.pdf)|*.pdf";
if (openFileDialog.ShowDialog() == DialogResult.OK)
{
    string pdfFilePath = openFileDialog.FileName;

    // Show Print dialog
    PrintDialog printDialog = new PrintDialog();
    printDialog.Document = new PrintDocument();
    printDialog.UseEXDialog = true;

    if (printDialog.ShowDialog() == DialogResult.OK)
```

```
            {
                // Print PDF file
                PrintDocument printDocument = printDialog.Document;
                printDocument.PrintController = new
StandardPrintController();
                printDocument.DefaultPageSettings.PaperSize = new
PaperSize("A4", 826, 595);
                printDocument.Print();
            }
        }
```

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм и теоретическую объектную модель решения задачи.
3. Какое событие отвечает за клик по кнопке?

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Перечислите все известные Вам библиотеки и компоненты для создания отчетов.
2. Приведите два способа установки библиотек и компонентов в Visual Studio.
3. Что означает запись not null при создании таблицы БД?

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.