Лабораторная работа 5

СОЗДАНИЕ КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ ДЛЯ ПРОСМОТРА, РЕДАКТИРОВАНИЯ ДАННЫХ БД.

ВЫЗОВ ХРАНИМЫХ ПРОЦЕДУР ИЗ КЛИЕНТСКОЙ ЧАСТИ

СОЗДАНИЕ ОТЧЕТНЫХ ФОРМ В КЛИЕНТСКОМ ПРИЛОЖЕНИИ

Цель работы – научиться создавать клиентское приложение для работы с базой данных с применением встроенных инструментов на Visual C#, научиться создавать формы отчетных документов по данным БД

Содержание работы:

1. Проработка примеров по ходу лабораторной работы.
2. Выполнение индивидуальных заданий.

Пояснения к выполнению работы

Для создания клиентского приложения на Visual C# (примеры приводятся для версии Visual C# 2017) используем пример базы данных **Library**, рассмотренную в лекциях. Реализуем следующие сценарии:

- просмотр книг в библиотеке с указанием общего количества экземпляров каждого наименования и количество выданных;
- поступление книг;
- выдача книг;
- просмотр, добавление, редактирования и удаление читателей;
- список должников.

Каждый сценарий будем реализовывать в отдельном окне. Пользовательский интерфейс приложения – многодокументный (MDI).

В проекте инструментов Data Sources создаем новое подключение к БД типа DataBase (Рисунок 1).

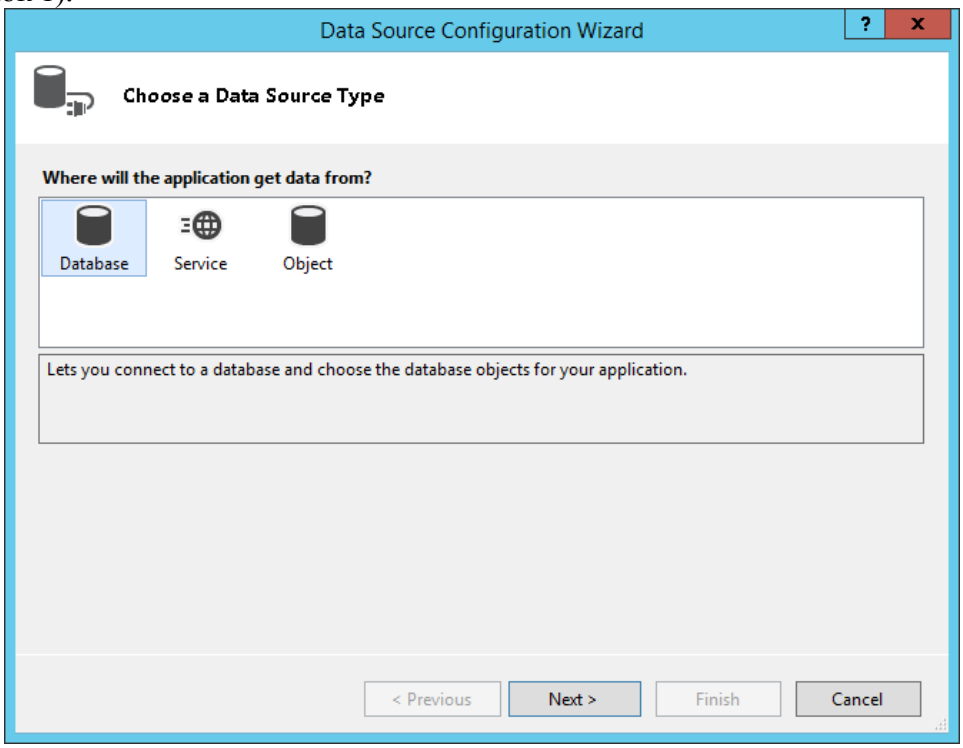

Рисунок 1 – Выбор источника данных

Следующим шагом выбираем модель БД – DataSet.

В окне подключения к БД (Рисунок 2) в поле Data Source выбираем Microsoft SQL

Server, в поле Server Name – **172.16.254.101** для подключения к учебному серверу в локальной сети университета или вашего локального сервера (обычно – localhost\SQLEXPRESS).

Тип аутентификации - SQL Server Authentication. Имя пользователя – stud, пароль – stud1.

Выбираем имя базы данных (Database name) - library.

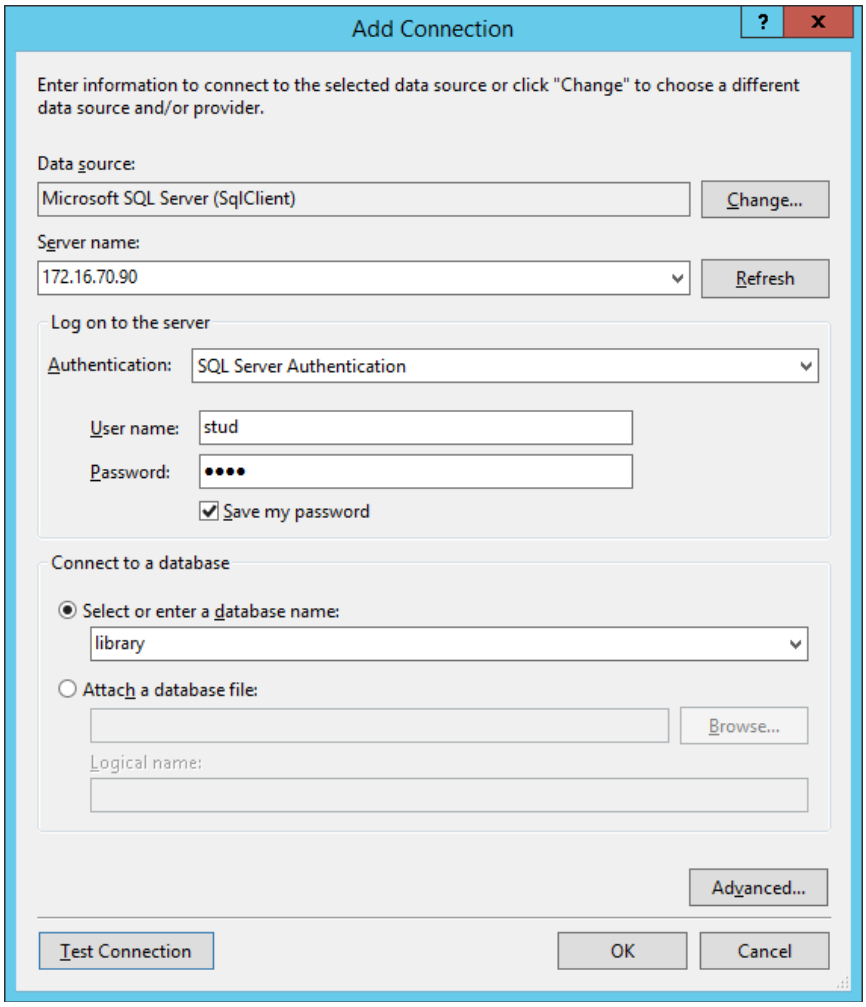

Рисунок 2 – Подключение к базе данных

На основной форме (Form1) добавить компонент MenuStrip. В редакторе меню создать пункты «Книги», «Читатели», «Должники».

Свойству IsMdiContainer основной формы присвоить значение True (Рисунок 3).

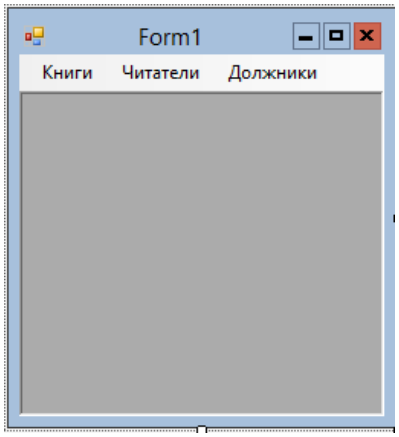

Рисунок 3 – Главная форма приложения

Создать формы: FReaders, FBooks, FDebtors.

Создать формы FNewBooks и FIssued. Две последние формы будут окнами диалога. Для придания им соответствующих атрибутов выставляем свойство FormBorderStyle в FixedDialog и добавим 2 кнопки Ok и Отмена со свойствами DialogResult – OK и Cancel.

На основной форме в пунктах меню в соответствующих методах Click вызвать соответствующие формы с помощью кода:

Для FBooks

```
Form f = new FBooks();
f.MdiParent = this;
f.Show();
```

Для FReaders, FDebtors – аналогично.

На форму FReaders «перетащить» с панели Data Sources таблицу READERS. При этом на форме должны создаться компоненты работы с данными:

1. libraryDataSet, который предоставляет доступ к кэшу данных и представляет полный набор данных, включая таблицы, содержащие, упорядочивающие и ограничивающие данные, а также связи между таблицами;
2. BindingSource, который предназначен для упрощения процесса привязки элементов управления к источнику данных и выступает в качестве источника и канала передачи данных для привязки других элементов управления;
3. TableAdapter: таблицы данных заполняются путем выполнения запросов к адаптеру таблицы TableAdapter или вызова команд адаптера обработки данных;
4. TableAdapterManager, который предоставляет функции сохранения данных в соответствующих таблицах данных;
5. BindingNavigator, который представляет собой пользовательский интерфейс для перехода и обработки для элементов управления, которые привязываются к данным; элемент управления BindingNavigator позволяет пользователям просматривать и управлять данными в форме Windows Forms;
6. DataGridView – для отображения и изменения табличных данные из различных типов источников данных.

Подобный набор компонентов часто используется для редактирования таблиц БД.

Обратите внимание, что в функции, обрабатывающей событие открытия формы Load должна присутствовать строка `this.rEADERSTableAdapter.Fill(this.libraryDataSet.READERS);` а в обработке нажатия кнопки сохранения:

```
this.Validate();
this.rEADERSBindingSource.EndEdit();
this.tableAdapterManager.UpdateAll(this.libraryDataSet);
```

Если в дальнейшим вы будете вручную подключать компоненты связи с БД, подобные выражения придется добавлять самостоятельно.

Откройте задачи для DataGridView (значок треугольника в правом верхнем углу), выберите в меню «Edit Columns…», для столбца EDUCATION для свойства ColumnType укажите DataGridViewComboBoxColumn (Рисунок 4). В список Items введите построчно значения «Начальное Среднее Высшее», соответствующие ограничению поля EDUCATION в базе данных.

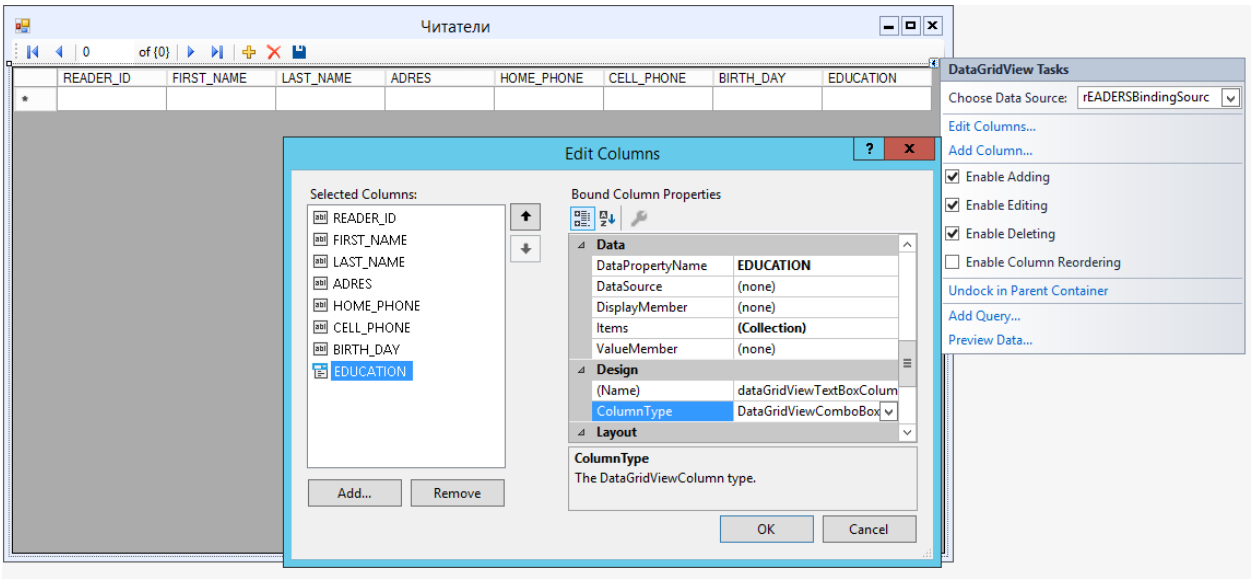

Рисунок 4 – Выбор из заданного списка значений

На форму FDebtors «перетащите» с панели Data Sources представление DEBTORS. На форме должны сформироваться такие же компоненты, как и на FReaders. Но так как сейчас мы работаем не с таблицей, а с представлением, представляющим собой соединение из нескольких таблиц, то редактировать данные на этой форме нельзя.

На форму FBooks «перетяните» с панели Data Sources представление exemplars. Тоже получим данные только для редактирования.

Создайте на панели навигатора кнопки для получения новых книг, выдачи и возврата читателям (Рисунок 5).

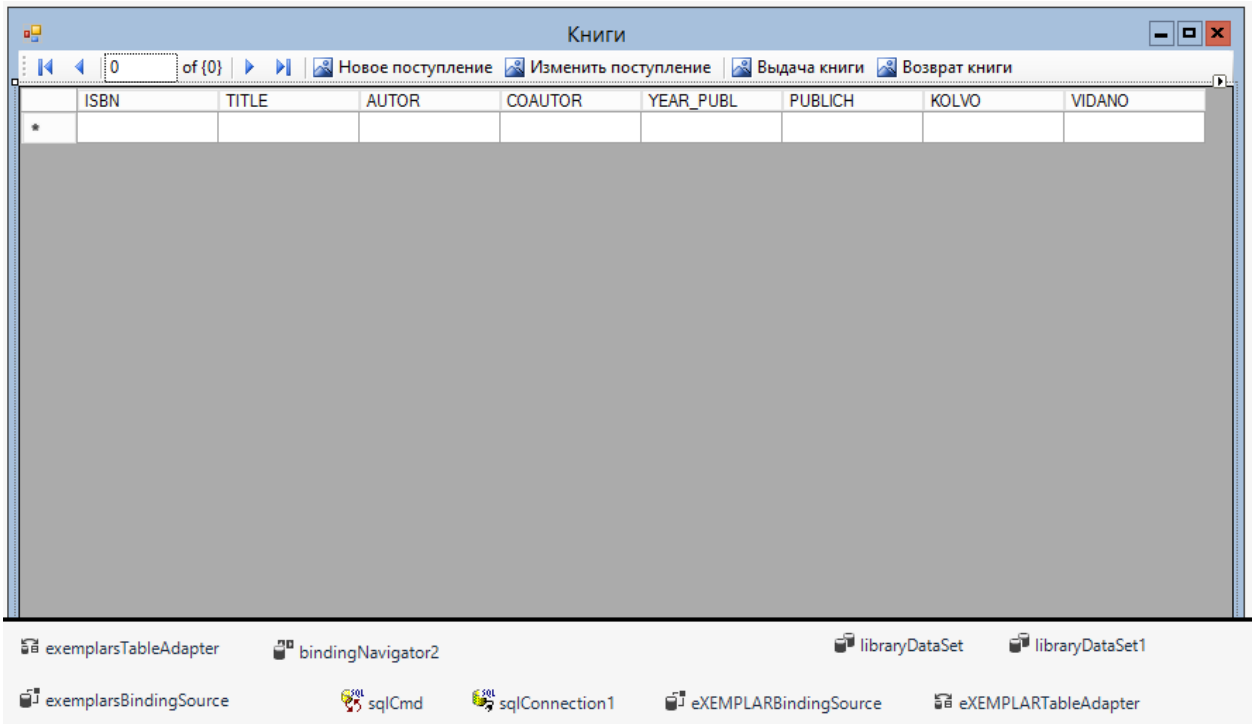

Рисунок 5 – Разработка формы FBooks

В обработке события кнопки «Новое поступление» введите следующий код:

```
FNewBook f = new FNewBook();
f.ShowDialog();
this.exemplarsTableAdapter.Fill(this.libraryDataSet.exemplars);
```

Для «Выдача книги»:

```
FIssued f = new FIssued();
f.ShowDialog();
this.exemplarsTableAdapter.Fill(this.libraryDataSet.exemplars);
```

Поместите на форму FNewbooks несколько компонентов Label и TextBox по количеству полей в таблице BOOKS (Рисунок 6).

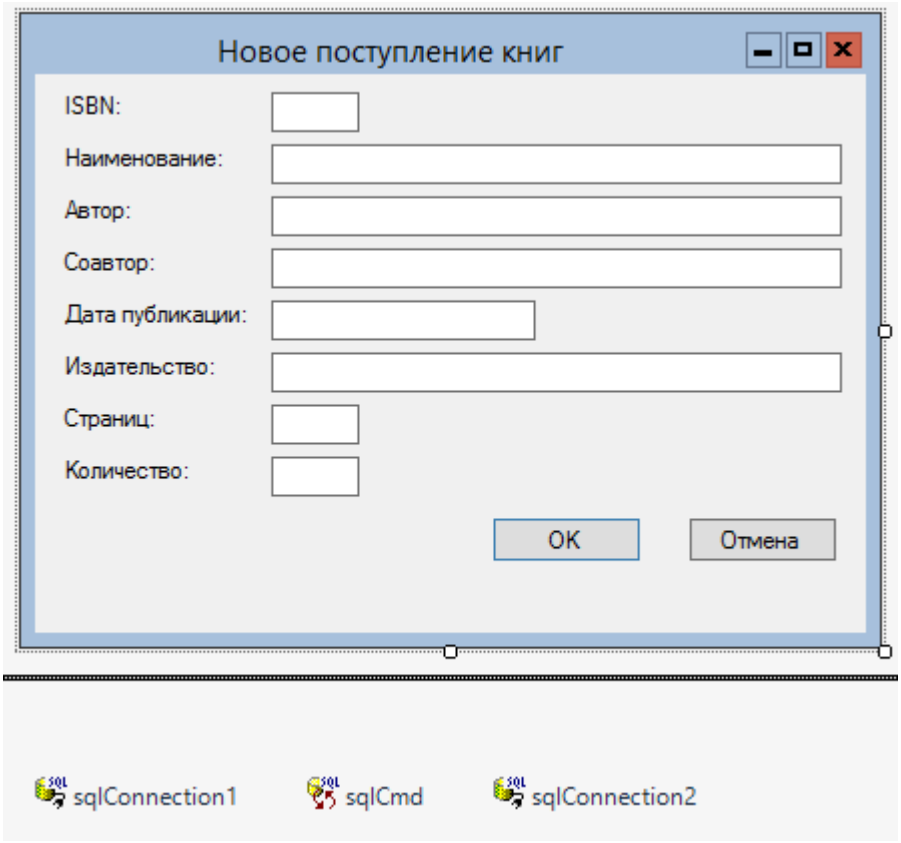

Рисунок 6 – Разработка формы FNewbooks

Теперь создадим код для выполнения хранимой процедуры по внесению новых книг.

Зайти Tools -> Choose Toolbox Items. Поставить отметки на элементах SqlCommand и SqlConnection, применить изменения.

Добавить на форму FNewBook компонент SqlConnection и в свойстве ConnectionString выбрать сформированную в начале работы строку подключения.

Теперь можно подключить хранимую процедуру NEW_BOOKS.

На форму добавить компонент SqlCommand (Рисунок 6). Изменить следующие его свойства:

Connection на SqlConnection1;

CommandType на StoredProcedure;

CommantText на NEW_BOOKS.

Создайте обработчик события FormClosing для формы FNewBook с текстом:

```
if (this.DialogResult == DialogResult.OK)
{
    sqlCmd.Parameters.AddWithValue("@ISBN", textBox1.Text);
    sqlCmd.Parameters.AddWithValue("@TITL", textBox2.Text);
    sqlCmd.Parameters.AddWithValue("@AUTOR", textBox3.Text);
    sqlCmd.Parameters.AddWithValue("@COAUTOR",
Convert.IsDBNull(textBox4.Text) ? Convert.DBNull : textBox4.Text);
    sqlCmd.Parameters.AddWithValue("@YEARIZD", textBox5.Text);
    sqlCmd.Parameters.AddWithValue("@PUBLICH", textBox6.Text);
    sqlCmd.Parameters.AddWithValue("@PAGES", textBox7.Text);
    sqlCmd.Parameters.AddWithValue("@NUM_EXEMPL", textBox8.Text);
    sqlConnection1.Open();
```

```
            sqlCmd.ExecuteNonQuery();
            sqlConnection1.Close();
        }
```

Теперь запрограммируем выдачу книг читателям.

Добавьте на форму FIssued компонент SqlConnection и в свойстве ConnectionString выбрать сформированную в начале работы строку подключения.

На форму добавить компонент SqlCommand. Изменить следующие его свойства:

Connection на SqlConnection1;

CommandType на StoredProcedure;

CommantText на NEW_INOUT.

Поместите на форму FIssued компоненты Label, ComboBox, dateTimePicker (Рисунок 7).

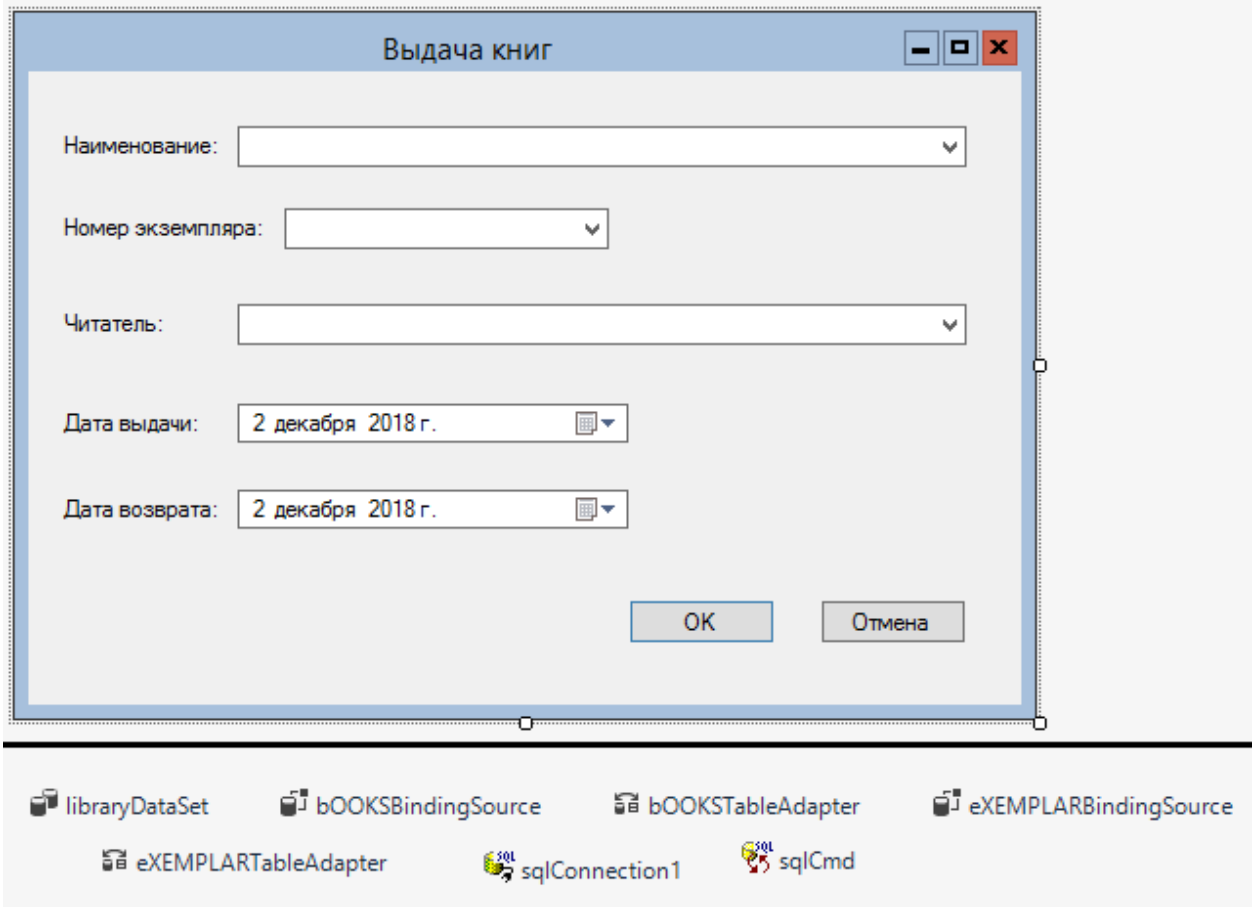

Рисунок 7 – Разработка формы FIssued

Настроить выпадающий список для читателя как показано на Рисунок 8. Теперь читатели берутся из БД.

Аналогичным образом настроить списки для книг и номеров экземпляров (Рисунок 9)

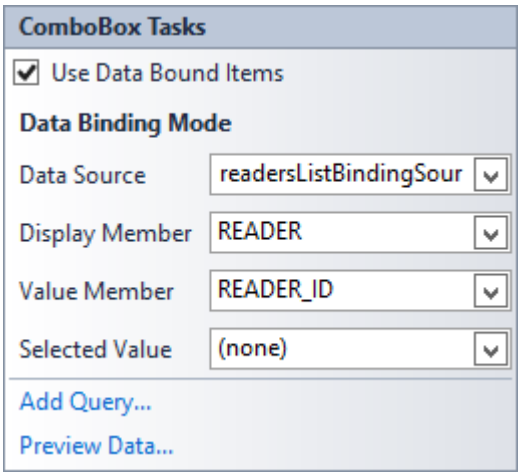

Рисунок 8 – Настройка списка читателей

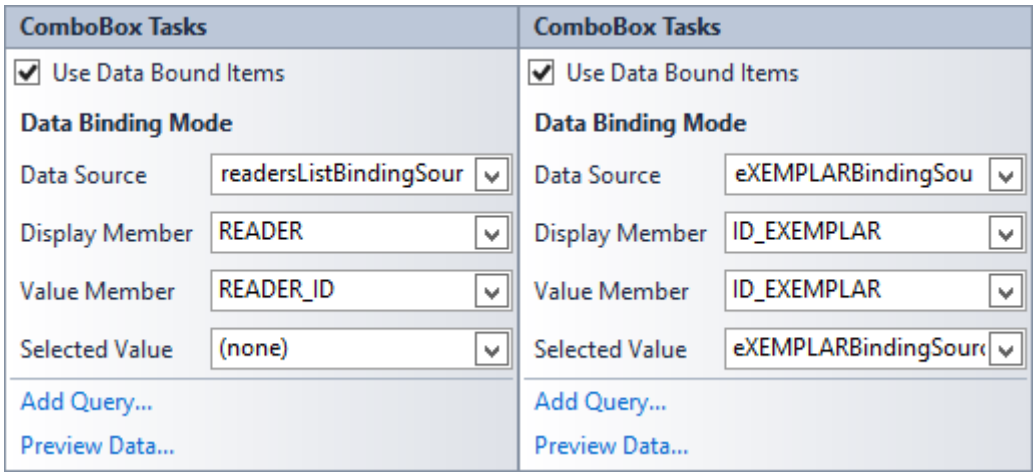

Рисунок 9 – Настройка списков наименований и номеров экземпляра

Для того, чтобы список экземпляров книг менялся при смене книги, переопределите событие TextChanged для comboBox1 (Наименование книги) со следующим кодом функции:

```
this.eXEMPLARTableAdapter.FillBy2(this.libraryDataSet.EXEMPLAR,
comboBox1.SelectedValue.ToString());
```

Убедитесь, что в коде открытия формы Load содержатся строки:

```
            // TODO: This line of code loads data into the 'libraryDataSet.Readers_List'
table. You can move, or remove it, as needed.
            this.readers_ListTableAdapter.Fill(this.libraryDataSet.Readers_List);
            // TODO: This line of code loads data into the 'libraryDataSet.EXEMPLAR'
table. You can move, or remove it, as needed.
            this.eXEMPLARTableAdapter.Fill(this.libraryDataSet.EXEMPLAR);
            // TODO: This line of code loads data into the 'libraryDataSet.BOOKS' table.
You can move, or remove it, as needed.
            this.bOOKSTableAdapter.Fill(this.libraryDataSet.BOOKS);
            this.eXEMPLARTableAdapter.FillBy2(this.libraryDataSet.EXEMPLAR,
comboBox1.SelectedValue.ToString());
```

Создайте обработчик события FormClosing для формы с текстом:

```
            if (this.DialogResult == DialogResult.OK)
            {
                sqlCmd.Parameters.AddWithValue("@ID_EXEMPLAR", comboBox2.Text);
                sqlCmd.Parameters.AddWithValue("@READER_ID",
comboBox3.SelectedValue.ToString());
                sqlCmd.Parameters.AddWithValue("@DATA_IN", dateTimePicker1.Value);
                sqlCmd.Parameters.AddWithValue("@DATA_OUT", dateTimePicker2.Value);
                sqlCmd.Parameters.AddWithValue("@EXIST", "0");
                sqlConnection1.Open();
                sqlCmd.ExecuteNonQuery();
                sqlConnection1.Close();
```

```
        }

    }
```

Проверьте работу приложения.

Отчеты во многом похожи на формы и тоже позволяют получить результаты работы запросов в наглядной форме, но только не на экране, а в виде распечатки на принтере. Таким образом, в результате работы отчета создается *бумажный документ*.

На Visual C# есть несколько способов создания отчетов. Один из способов создание отчетов — это использование генератора отчета FastReport. Генератор можно скачать с официального сайта компании https://www.fast-report.com/ru/download/. После установки генератора необходимо перезапустить Visual C#. Затем необходимо добавить компоненты FastReport.

Для создания отчета необходимо поместить компонент Report на главную форму. После этого двойным щелчком нажать на компонент и выбрать данные, которые нам нужны для составления отчета. После этого откроется сам редактор отчетов. Для сохранения изменений нужно просто сохранить файл отчета в любом месте.

Структура отчета

Отчеты состоят из разделов или секций (Bands), а разделы могут содержать элементы управления. Для настройки разделов надо нажать на рабочей области на кнопку «Настроить бэнды».

1. Структура отчета состоит из следующих разделов: заголовок отчета, подвал отчета, заголовок страницы, подвал страницы, область данных, заголовок колонки, подвал колонки, фоновый.
2. Раздел ***заголовок отчета*** служит для печати общего заголовка отчета.
3. Раздел ***заголовок страницы*** можно использовать для печати подзаголовков, если отчет имеет сложную структуру и занимает много страниц.
   Здесь можно также помещать и номера страниц, если это не сделано в нижнем колонтитуле.
4. В ***области данных*** размещают элементы управления, связанные с содержимым полей таблиц базы. В эти элементы управления выдаются данные из таблиц для печати на принтере. Эти разделы будут на печати воспроизводиться столько раз, сколько записей присутствует в привязанном запросе или таблице.
5. Раздел ***подвал страницы*** используют для тех же целей, что и раздел заголовок страницы. Можно использовать для подстановки полей для подписей должностных лиц, если есть необходимость подписывать отчет на каждой странице.
6. Разделы ***колонок*** используют для размещения дополнительной информации или итоговой информации по всем данным отчета. Печатается сверху или снизу области данных.
7. Для предварительного просмотра отчета в том виде, как он будет расположен на бумаге, необходимо вызвать метод **Show** компонента **Report (**на главной форме в меню добавить раздел и в методе **Click** написать этот метод, например, **Report1.Show())**. Пример отчета в режиме «Конструктор» представлен на рис 2, а в режиме предварительного просмотра – на рис. 3.

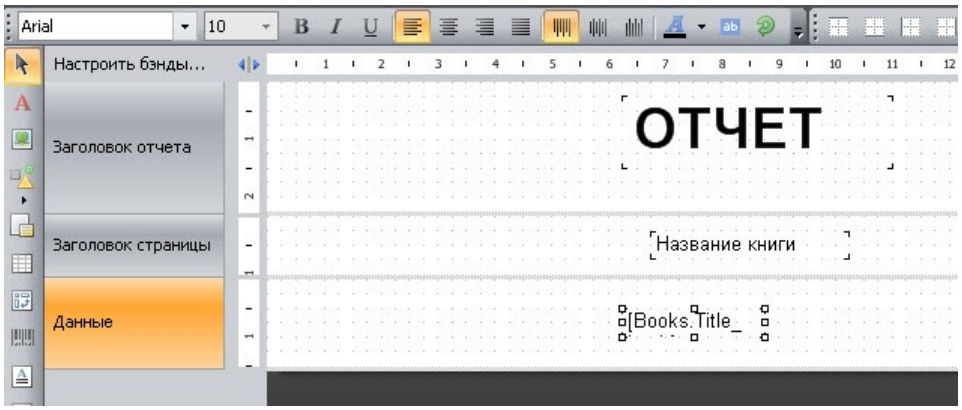

Рис 2. Пример отчета в режиме «Конструктор»

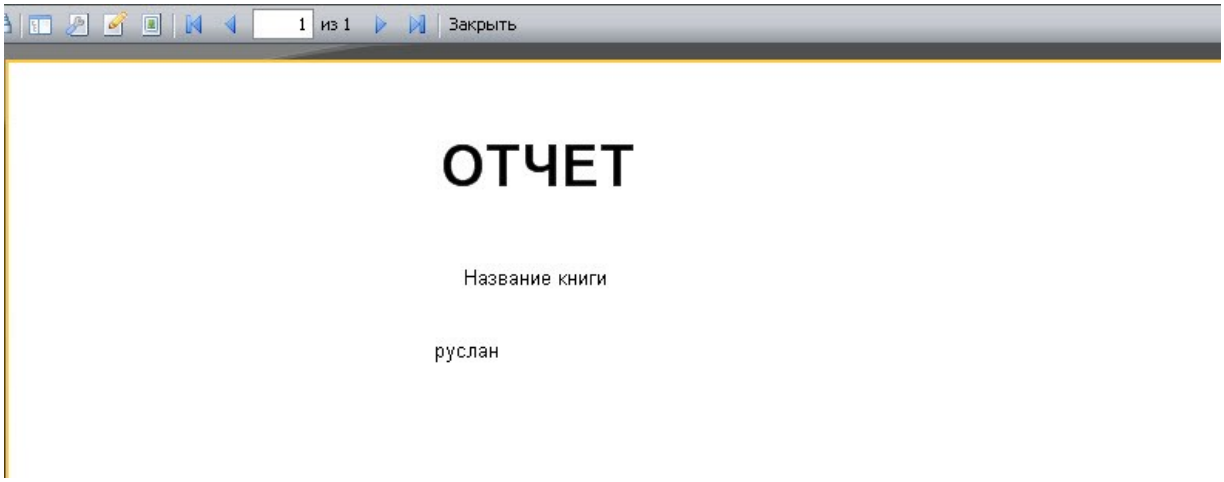

Рис. Пример отчета в предварительном просмотре

Пример 1. Создание отчета в табличной форме, который выбирает из таблицы Books наименование, автор и издательство.

1. В проекте на главной форме в меню добавить пункт меню **Отчеты**,
   а также подпункты:
   Отчет в табличной форме;
   Отчет в свободной форме;
   Отчет с группировкой по двум таблицам.
2. В проекте на главную форму добавить 3 компонента **Report**
3. У первого компонента **Report** изменить **DataSet** (стрелка на компоненте Task -> Select DataSet) на соответствующие данные необходимые для отчета. Открыть окно дизайна отчета (двойным щелчком по компоненту).
4. В свойствах включить такие разделы (настроить бэнды), как ***заголовок отчета, заголовок данных, данные***.
5. В разделе **заголовок отчета** разместить метку (компонент **Текст**). В свойствах изменить его внешний вид и подпись «Пример табличного отчета».
6. В разделе **заголовок данных** установить компонент **Таблица** (для имитации обрамления шапки таблицы) и написать в ней **Название книги, Автор, Издательство**.
7. На раздел **данные** перетянуть объекты с панели *данные* по следующему пути:
   Источники данных -> Books -> title
   Источники данных -> Books -> Autor
   Источники данных -> Books -> Publish
   Расположить компоненты симметрично под надписями в таблице.

8. В главной форме приложения в подпункте **Отчет в табличной форме** в методе Click написать команду: **Report1.Show().**
9. Запустить приложение, проверить работу.

Пример 2. Создание отчета в свободной форме с данными из первого задания. Создадим карточку книги для библиотечной картотеки.

Особенность отчета в свободной форме в том, что он создает шаблон на каждую отдельную запись таблицы, другими словами, он создается по документам, у которых нет шапки и примечаний. Примером таких документов может служить приходный или расходный кассовый ордер, этикетка для товара или ценник в магазине, пригласительное письмо и т.д.

1. У второго компонента **Report** установить свойства **DataSet** на необходимые. В свойствах (настроить бэнды) включить раздел ***Данные***.
2. На раздел **данные** перетянуть объекты с панели *данные* по следующему пути:
   Источники данных -> Books -> title
   Источники данных -> Autors -> name_autor
   Источники данных -> Publishing_house -> publish
3. В главной форме приложения в подпункте **Отчет в свободной форме** в методе Click написать команду: **Report2.Show().**
4. Запустить приложение, проверить работу.

Пример 3. Создание отчета по двум таблицам. Создадим отчет с группировкой, в котором сначала будут выводиться данные читателя из представления Readers_List, а затем список книг, которые брал этот читаттель.

1. У третьего компонента **Report** установить свойства **DataSet** на необходимые. В свойствах (*настроить бэнды*) включить разделы: ***заголовок отчета, данные***.
2. В разделе **заголовок отчета** разместить метку (компонент **Текст**). В свойствах изменить ее внешний вид и подпись «Отчет по читателям и прочитанным книгам».
3. Вызвать мастер группировки. Панель *Отчёт -> Мастер группировки*. В качестве условия группировки указать поле, по которому будет осуществляться группировка данных: Readers -> LastName. Нажать *Добавить*.
4. В результате получим бэнды: Заголовок группы (содержит фамилию читателя), Данные, Подвал группы. На раздел **данные** перетенуть обьекты с панели *данные* по следующему пути: Источники данных -> Readers -> Exemplar ->ISBN.
5. В главной форме приложения в подпункте **Отчет с группировкой по двум таблицам** в методе Click написать команду: **Report3.Show().**
6. Запустить приложение, проверить работу.

Задания к лабораторной работе №6

1. Самостоятельно разработать форму редактирования информации о книге.
2. Разработать форму возврата книги читателем.
3. Создать отчет, которые показывает информацию о книгах, взятых читателем с их наименованиями и датами получения и возврата.
4. Создать отчет, который показывает 10 наиболее востребованных книг.
5. Создать отчет, который показывает 10 наименее востребованных книг.

Дополнительное задание. На Visual C# создать проект для индивидуальной БД, созданной в предыдущих лабораторных работах, создать интерфейс, аналогичный описанному по ходу текущей лабораторной работы. Результаты запросов поиска вывести в виде отчетов.