**Министерство науки и высшего оразования РФ**

**ФГБОУ ВО  
«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**Кафедра «Информатика и программное обеспечение»**

**Расчетно-графическая работа**

**по дисциплине «Программирование в среде Microsoft .NET»**

направление подготовки 09.03.04 «Программная инженерия»

на тему:

**Разработка приложения для автоматизированного сбора, хранения и обработки информации о банках**

Выполнил студ. гр. О-18-ПРИ-рпс-Б:

**Подгорняк А. А.**

Проверил:

к.т.н., доц. **Коростелев Д. А.**

**Брянск 2021**

# ЗАДАНИЕ

В данной расчетно-графической работе необходимо разработать приложение Windows Forms для автоматизированного сбора, хранения и обработки данных информации о продуктах с выбранного веб-ресурса.

Поставленные задачи:

1. Разработать приложение Windows Forms, которое позволяет загружать в автоматизированном режиме информацию о продуктах с внешнего источника в БД.
2. Для хранения загруженной информации в программе необходима БД, структуру которой также необходимо разработать.
3. Программа должна иметь меню и соответствующие элементы управления, позволяющие запускать процесс первоначальной загрузки информации с внешнего источника, а также осуществлять дозагрузку. При этом должно быть исключено дублирование информации. Вся загружаемая информация должна помещаться в БД.
4. Программа дополнительно должна иметь элементы управления, позволяющие просматривать, редактировать и удалять загруженные записи из БД, а также позволять добавлять новые записи.
5. Программа должна быть хорошо протестирована, а также должны быть предусмотрены конструкции обработки исключительных ситуаций.

Текущая расчетно-графическая работа направлена на закрепление и развитие навыков разработки многофункциональных приложений для платформы Microsoft .Net.

# СОДЕРЖАНИЕ

[ЗАДАНИЕ 2](#_Toc69713711)

[СОДЕРЖАНИЕ 3](#_Toc69713712)

[1. ВЫБОР ИСТОЧНИКА ИНФОРМАЦИИ И МЕТОДА РАБОТЫ С НИМ 4](#_Toc69713713)

[1.1. Описание и обоснование выбора источника информации 4](#_Toc69713714)

[1.2. Описание и обоснование метода извлечения данных из источника 5](#_Toc69713715)

[2. АРХИТЕКТУРА РАЗРАБОТАННОЙ ПРОГРАММЫ 6](#_Toc69713716)

[3. ОПИСАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ 7](#_Toc69713717)

[4. ОПИСАНИЕ ИЗВЛЕЧЕНИЯ ДАННЫХ С ВНЕШНЕГО ИСТОЧНИКА 8](#_Toc69713718)

[5. ОПИСАНИЕ ИНТЕРФЕЙСА РАЗРАБОТАННОЙ ПРОГРАММЫ 12](#_Toc69713719)

[6. ОПИСАНИЕ ПРОЦЕССА ТЕСТИРОВАНИЯ ПРОГРАММЫ 14](#_Toc69713720)

[ЗАКЛЮЧЕНИЕ 15](#_Toc69713721)

[СПИСОК ЛИТЕРАТУРЫ 16](#_Toc69713722)

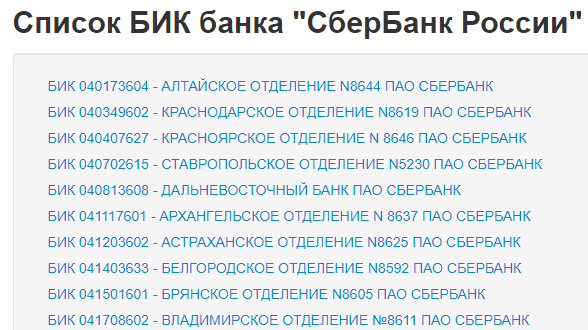
# ВЫБОР ИСТОЧНИКА ИНФОРМАЦИИ И МЕТОДА РАБОТЫ С НИМ

## Описание и обоснование выбора источника информации

При выборе источника информации преследовалось условие – информация должна быть каким-либо образом структурирована, поскольку извлечь конкретные данные из сплошного текста достаточно сложно, а чаще всего вообще невозможно.

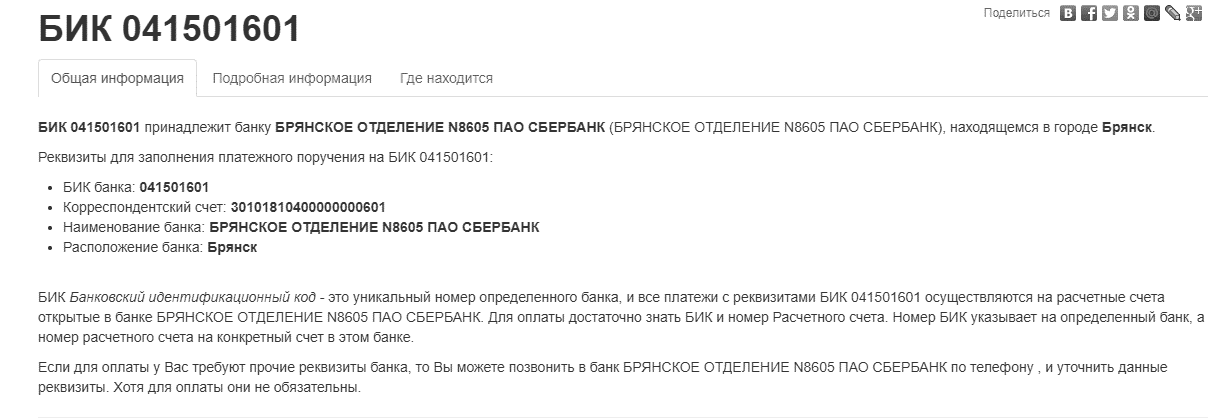
Из изученных веб-ресурсов наиболее подходящим оказался <https://bik-info.ru>,

так как там информация удобно структурирована.



**Рис. 1. Внешний вид источника информации**

В данном источнике вся необходимая информация представлена в виде списка, разделенного по категориям. Для каждого банка указано его название, номер отделения, а также территориальное расположение. При нажатии на банк можно перейти на отдельную веб-страницу с его подробным описанием.



**Рис. 2. Отдельная страница банка**

HTML-представление любого продукта имеет однотипную структуру, которая представлена на листинге 1.

*Листинг 1*

|  |
| --- |
| <li>  <a href="/bik\_041501601.html" title="Бик 041501601 из Брянск">БИК 041501601 - БРЯНСКОЕ ОТДЕЛЕНИЕ N8605 ПАО СБЕРБАНК</a>  </li> |

Листинг 1. HTML-код отдельного банка в списке

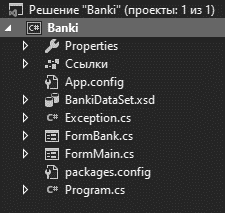
## Описание и обоснование метода извлечения данных из источника

В качестве метода, с помощью которого информация будет собираться, был выбран синтаксический разбор html-страниц. При этом синтаксический разбор и навигацию по html-страницам программа будет осуществлять с помощью библиотека HtmlAgilityPack.

Выбор пал на данную библиотеку, потому что это один из самых известный парсер HTML в мире .NET. HtmlAgilityPack быстрая, относительно удобная библиотека для работы с HTML. Для выбора элементов используется язык запросов XPath.

# АРХИТЕКТУРА РАЗРАБОТАННОЙ ПРОГРАММЫ

Архитектура разработанной программы подходит под описание трехуровневой модели. Она представлена на рис. 3.



**Рис. 3. Архитектура программы**

Три уровня (с позиции программирования) – это хранение, обработка и представление информации. Идея заключается в том, чтобы не смешивать эти три составляющие. Новизна же и идея распределенных приложений состоит в том, чтобы иметь возможность распределить эти уровни физически на различных компьютерах, а также возможность иметь несколько взаимозаменяемых вариантов каждого уровня.

Первый уровень – уровень представления – реализован с помощью компонентов Windows Forms, позволяющих создать интерфейс, привычный и понятный пользователю. Кроме того, они позволяют отобразить данные в удобочитаемом формате.

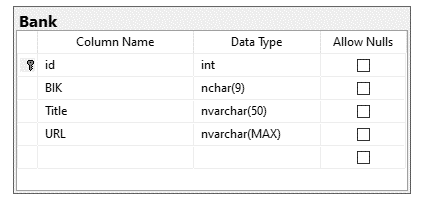
Второй уровень – уровень логики приложения – реализован с помощью DataSet. После получения данных из базы данных через DataSet мы можем локально работать с этими данными вне зависимости от наличия подключения.

Третий уровень – уровень доступа к данным – реализован с помощью СУБД MS SQL Server, где была создана база данных, связь с которой осуществляется с помощью DataSet.

# ОПИСАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

Для хранения информации использовалась СУБД Microsoft SQL Server, являющаяся одной из наиболее популярных систем управления базами данных в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

С помощью графических средств SQL Management Studio была создана база данных Banki, включающую в себя таблицу Bank (рис. 4).



**Рис. 4. Структура базы данных**

Каждая запись таблицы Bank содержит:

* id – уникальный идентификатор, который автоматически увеличивается на единицу для каждой новой записи;
* Title – название банка, которое может содержать максимум 50 символов;
* URL – ссылка на страницу банка;
* BIK – Банковский идентификационный код.

# ОПИСАНИЕ ИЗВЛЕЧЕНИЯ ДАННЫХ С ВНЕШНЕГО ИСТОЧНИКА

При запуске программы, обработчик загрузки главной формы вызывает функцию Parsing().Сначала мы загружаем код страницы. Затем обрабатываем список банков и добавляем информацию в БД, если она не повторяется.

*Листинг 2*

|  |
| --- |
| static string LoadPage(string url)  {  var result = "";  var request = (HttpWebRequest)WebRequest.Create(url);  var response = (HttpWebResponse)request.GetResponse();  if (response.StatusCode == HttpStatusCode.OK)  {  var receiveStream = response.GetResponseStream();  if (receiveStream != null)  {  StreamReader readStream;  if (response.CharacterSet == null)  readStream = new StreamReader(receiveStream);  else  readStream = new StreamReader(receiveStream, Encoding.GetEncoding(response.CharacterSet));  result = readStream.ReadToEnd();  readStream.Close();  }  response.Close();  }  return result;  } |

Листинг 2. Обработчик загрузки кода страницы

*Листинг 3*

|  |
| --- |
| void Parsing()  {  //загружаем страницу  var pageContent = LoadPage(@"https://bik-info.ru/name\_1.html");  var document = new HtmlAgilityPack.HtmlDocument();  document.LoadHtml(pageContent);  var classValue = "nav nav-list";  var LIst = document.DocumentNode.SelectNodes(".//\*[@class='" + classValue + "']/li") ?? Enumerable.Empty<HtmlNode>();  //добавление в БД  foreach (HtmlNode li in LIst)  {  DataRow Row = bankiDataSet.Bank.NewRow();  var html = li.InnerHtml;  var text = li.InnerText;  string[] stringSeparators = new string[] { " - " };  string[] words = text.Split(stringSeparators, StringSplitOptions.None);  string[] BIK = words[0].Split(' ');  Regex regex = new Regex(@"\w\*.html\w\*");  MatchCollection matche = regex.Matches(html);  Row[1] = BIK[1];  Row[2] = words[1];  Row[3] = "bik-info.ru/" + matche[0].Value;  if(bankiDataSet.Bank.Select("BIK=" + Row[1]).Length==0)  bankiDataSet.Bank.Rows.Add(Row);  }  bankTableAdapter.Update(bankiDataSet.Bank);  bankiDataSet.AcceptChanges();  BankiDataGridView.DataSource = bankBindingSource;  } |

Листинг 3. Обработка и добавление информации

С помощью языка запросов XPath находятся все элементы списка и сохраняются в переменной LIst. Затем эти элементы обрабатываются при помощи разделения строки по разделителю, а для URL адреса используется регулярное выражение. В пустой экземпляр строки таблицы записывается информация, и перед добавлением в БД осуществляется проверка на повторение.

Кроме этого, в моей программе присутствуют кнопки для редактирования, добавления, удаления записей.

*Листинг 4*

|  |
| --- |
| private void buttonDeleteBank\_Click(object sender, EventArgs e)  {  var rowIndex = this.BankiDataGridView.SelectedCells[0].RowIndex;  BankiDataGridView.Rows.RemoveAt(rowIndex);  bankiDataSet.Tables["Bank"].Rows[rowIndex].Delete();  bankTableAdapter.Update(bankiDataSet);  }  private void buttonAddBank\_Click(object sender, EventArgs e)  {  var bank = new FormBank(this);  bank.ShowDialog();  }  private void buttonEditBank\_Click(object sender, EventArgs e)  {  if (BankiDataGridView.SelectedRows.Count > 0)  {  int selectedrowindex = BankiDataGridView.SelectedCells[0].RowIndex;  DataGridViewRow selectedRow = BankiDataGridView.Rows[selectedrowindex];  var bank = new FormBank(selectedRow, this);  bank.ShowDialog();  this.bankTableAdapter.Fill(this.bankiDataSet.Bank);  }  } |

Листинг 4. Обработчики на главной форме

Для создания и редактирования записей реализована отдельная форма в виде popup окна. При изменении записи на дополнительную форму передаются данные выделенной строки из таблицы DataGridView основной формы.

*Листинг 5*

|  |
| --- |
| public FormBank(DataGridViewRow selectedRow, FormMain frm)  {  Size Screen = SystemInformation.PrimaryMonitorSize;  this.Location = new Point(Screen.Width - Width, Screen.Height - Height);  InitializeComponent();  this.form = frm;  textBoxBIK.Text = selectedRow.Cells[1].Value.ToString();  textBoxName.Text = selectedRow.Cells[2].Value.ToString();  textBoxURL.Text = selectedRow.Cells[3].Value.ToString();  select = selectedRow.Index;  buttonAdd.Visible = false;  buttonEdit.Visible = true;  } public FormBank(FormMain frm)  {  Size Screen = SystemInformation.PrimaryMonitorSize;  this.Location = new Point(Screen.Width - Width, Screen.Height - Height);  this.form = frm;  InitializeComponent();  buttonAdd.Visible = true;  buttonEdit.Visible = false;  }  private void buttonAddBank\_Click(object sender, EventArgs e)  {  try  {  if (string.IsNullOrWhiteSpace(textBoxBIK.Text) || string.IsNullOrWhiteSpace(textBoxName.Text) || string.IsNullOrWhiteSpace(textBoxURL.Text))  throw new Banki.Exception("Данные заполнены некорректно!");  DataRow Row = form.bankiDataSet.Bank.NewRow();  Row[1] = textBoxBIK.Text;  Row[2] = textBoxName.Text;  Row[3] = textBoxURL.Text;  form.bankiDataSet.Bank.Rows.Add(Row);  form.bankTableAdapter.Update(form.bankiDataSet.Bank);  form.bankiDataSet.AcceptChanges();  form.BankiDataGridView.DataSource = form.bankBindingSource;  this.Close();  }  catch (System.Exception exception)  {  MessageBox.Show(exception.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);  }  }    private void buttonEditBank\_Click(object sender, EventArgs e)  {  DataRowView drw = (DataRowView)form.bankBindingSource.Current;  BankiDataSet.BankRow row = (BankiDataSet.BankRow)drw.Row;  row.BIK = textBoxBIK.Text;  row.Title = textBoxName.Text;  row.URL = textBoxURL.Text;  if (select != -1)  {  form.bankiDataSet.Bank.Rows[select].ItemArray[1] = row.ItemArray[1];  form.bankiDataSet.Bank.Rows[select].ItemArray[2] = row.ItemArray[2];  }  else  form.bankiDataSet.Bank.Rows.Add(row);  form.tableAdapterManager.UpdateAll(form.bankiDataSet);  form.bankiDataSet.AcceptChanges();  this.Close();  } |

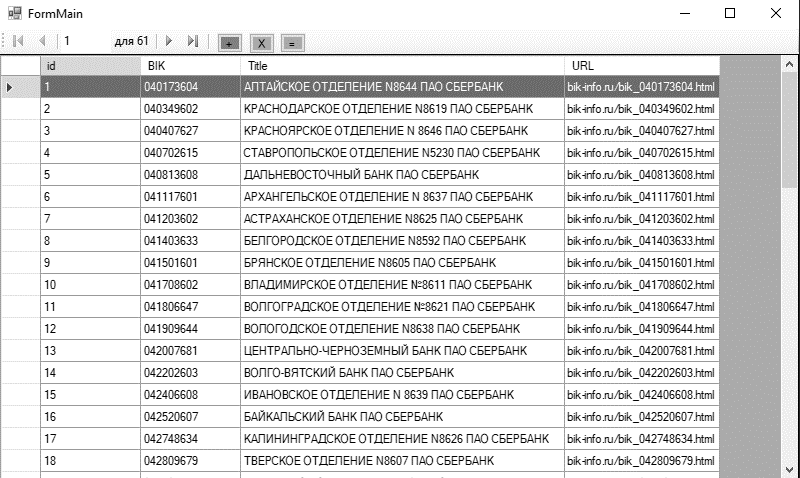
Листинг 5. Обработчики на второй форме

При добавлении записи осуществляется проверка на пустые поля, и в случае положительного результата программа выбрасывает исключение с описанием ошибки.

# ОПИСАНИЕ ИНТЕРФЕЙСА РАЗРАБОТАННОЙ ПРОГРАММЫ

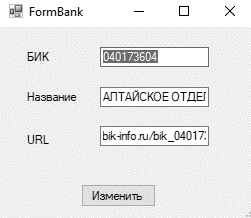
Для создания интерфейса в расчётно-графической работе использовалось приложение Windows Forms, в котором была создана главная форма приложения (рис. 5), включающая в себя представленный список компонентов:

* BindingNavigator, представляющий собой пользовательский интерфейс для перехода и обработки для элементов управления на форме, которые привязываются к данным.
* DataGridView – класс, отображающий данные в настраиваемой сетке (таблице). Сюда, аналогично отображению во внешнем источнике, в качестве строк выводится информация о каждом продукте: его название, адрес собственной страницы, категория и состояние.



**Рис. 5. Интерфейс приложения**

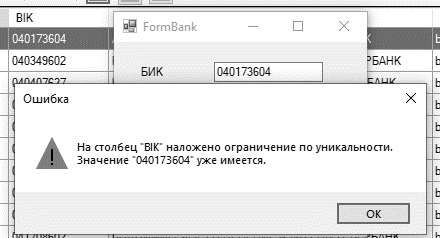
После загрузки данных с веб-сайта приложение принимает вид, показанный на рис. 5.



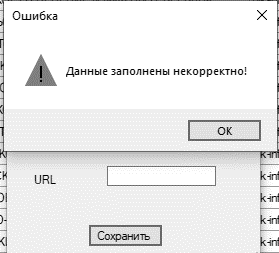
**Рис. 6. PopUp окно для изменения**

# ОПИСАНИЕ ПРОЦЕССА ТЕСТИРОВАНИЯ ПРОГРАММЫ

При попытке добавить в таблицу повторяющееся или пустое значение приложение обработает эту ситуацию и выдаст сообщение об ошибке.



**Рис. 9. Попытка добавить в таблицу неуникальное значение**



**Рис. 10. Попытка добавить в таблицу пустое значение**

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной расчётно-графической работы было создано приложение Windows Forms, позволяющее загружать информацию с веб-сайта по требованию пользователя и сохранять полученную информацию в базу данных, а также просматривать, редактировать, добавлять и удалять информацию по требованию пользователя БД.

За время разработки и тестирования было выяснено, что программа работает без критических ошибок и полностью соответствует требованиям задания.

В результате проделанной работы были получены важные навыки работы с базами данных, пользовательским интерфейсом приложений и парсингом информации.

# СПИСОК ЛИТЕРАТУРЫ

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж.Рихтер. – СПб.: Питер, 2016. – 896 с.
2. Троелсен, Э. Язык программирования С# 5.0 и платформа .NET 4.5 / Э.Троелсен. – 6-е изд. – М.: Вильямс, 2015. – 1312 с.
3. Неш, Т. C# 2015: ускоренный курс для профессионалов: [пер. с англ.] / Т.Неш – М.: Вильямс, 2015. – 576 с.
4. Нейгел, К. C# 5.0 и платформа .NET 4.5 для профессионалов / К.Нейгел, Б.Ивьен, Дж.Глинн, М.Скиннер, К.Уотсон. – М.: Диалектика, 2016. – 1440с.
5. Официальная документация CsQuery [Электронный ресурс]. – Режим доступа: <https://github.com/jamietre/CsQuery> свободный (19.12.2020).