**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 растениях**

Выполнил студ. гр. О-18-ПРИ-рпс-Б:

**Лядов В. С.**

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.

**Оглавление**

[ЗАДАНИЕ 2](#_Toc78237005)

[1. ВЫБОР ИСТОЧНИКА ИНФОРМАЦИИ И МЕТОДА РАБОТЫ С НИМ 4](#_Toc78237006)

[1.1. Описание и обоснование выбора источника информации 4](#_Toc78237007)

[1.2. Описание и обоснование метода извлечения данных из источника 7](#_Toc78237008)

[2. АРХИТЕКТУРА РАЗРАБОТАННОЙ ПРОГРАММЫ 9](#_Toc78237009)

[3. ОПИСАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ 11](#_Toc78237010)

[4. ОПИСАНИЕ ИЗВЛЕЧЕНИЯ ДАННЫХ С ВНЕШНЕГО ИСТОЧНИКА 12](#_Toc78237011)

[5. ОПИСАНИЕ ИНТЕРФЕЙСА РАЗРАБОТАННОЙ ПРОГРАММЫ 18](#_Toc78237012)

[6. ОПИСАНИЕ ПРОЦЕССА ТЕСТИРОВАНИЯ ПРОГРАММЫ 20](#_Toc78237013)

[ЗАКЛЮЧЕНИЕ 22](#_Toc78237014)

[СПИСОК ЛИТЕРАТУРЫ 23](#_Toc78237015)

# 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 <http://floristika.info> , так как там информация имеет следующую иерархию.



**Рис. 1. Внешний вид главной страницы с категориями**

****

**Рис. 2. Внешний вид страницы с растениями**

****

**Рис. 3. Внешний вид пагинации для каждой категории.**

В данном источнике вся необходимая информация представлена в табличном виде. При нажатии на растение можно перейти на отдельную веб-страницу с его подробным описанием. Для каждого растения указано его название, категория, семейство, а также латинское название и другие называния.



**Рис. 4. Отдельная страница растения**

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

Листинг 1 HTML-код отдельного растения

|  |
| --- |
| <h1>Абутилон</h1>  <div class="js\_shop\_id js\_shop shop shop\_id shop-item-container">  <div class="shop-item-left">  <div class="js\_shop\_all\_img shop\_all\_img shop-item-big-images">  <a class="js\_shop\_img shop-item-image active" href="[<http://floristika.info>/userfls/shop/large/1\_abutilon.jpg](http://procweti.ru/userfls/shop/large/1_abutilon.jpg)" data-fancybox="gallery1shop" image\_id="1">  <img src="[<http://floristika.info>/userfls/shop/large/1\_abutilon.jpg](http://procweti.ru/userfls/shop/large/1_abutilon.jpg)" alt="Абутилон" title="Абутилон" image\_id="1" class="shop\_id\_img">  </a>  <span class="icon-zoom"></span>  </div>  <div class="shop\_preview\_img shop-item-previews items-scroller" data-item-per-screen="3" data-controls="shop-previews-control">  <a class="js\_shop\_preview\_img item" href="[#](http://procweti.ru/abytilon)" style="background-image:url(/userfls/shop/preview/1\_abutilon.jpg)" image\_id="1">  </a></div></div>  <div class="shop\_text">  <p><u>Латинское название:</u> Abutilon.</p>  <p><u>Семейство:</u> Мальвовые.</p>  <p><u>Семейство:</u> Мальвовые.</p>  <p>Родиной Абутилона являются тропики и субтропики. В культуре наиболее известным видом является Абутилон гибридный (Abutilon hybridum hort), получен данный комнатный цветок путем скрещивания нескольких видов. Сегодня его используют для озеленения жилых помещений. В дикой природе этот гибрид не встречается.</p> |

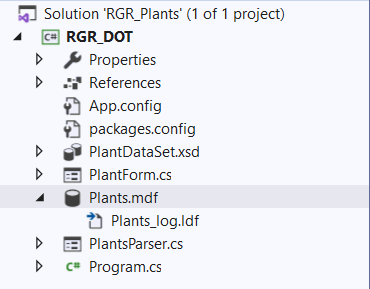
## Описание и обоснование метода извлечения данных из источника

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. 5.



**Рис. 5. 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.

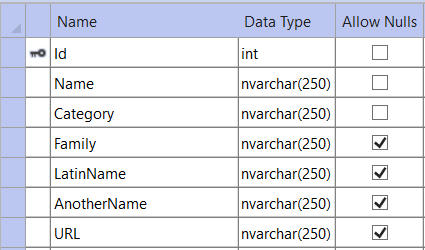
Второй уровень – уровень логики приложения – реализован с помощью DataSet. После получения данных из базы данных через DataSet мы можем локально работать с этими данными вне зависимости от наличия подключения.

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

# ОПИСАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

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

С помощью графических средств Visual Studio Community была создана база данных Plants, включающую в себя таблицу Plant (рис. 6).



**Рис. 6. Структура базы данных**

Каждая запись таблицы Plant содержит:

* id – уникальный идентификатор, который автоматически увеличивается на единицу для каждой новой записи;
* Name – название растения, которое может содержать максимум 250 символов;
* Category – категория, к которой относится данное растение;
* Family – семейство, к которой относится данное растение;
* LatinName – Латинское название растения;
* AnotherName – Другое имя растения;
* URL – ссылка на страницу банка;

# ОПИСАНИЕ ИЗВЛЕЧЕНИЯ ДАННЫХ С ВНЕШНЕГО ИСТОЧНИКА

При запуске программы, обработчик загрузки главной формы вызывает функцию Parsing(). Сначала мы загружаем код главной страницы, считываем ссылки категорий, загружаем код страницы 1 страницы с растениями, считываем ссылки на остальные страницы данной категории, переходя по каждой странице категории, переходим по ссылке на каждое растение для более полной информации о нем. Затем обрабатываем информацию о каждом растении и добавляем ее в БД, если она еще не встречалась.

Листинг 2 Обработчик загрузки кода страницы

|  |
| --- |
| 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 3 Обработка и добавление информации

|  |
| --- |
| void Parsing()  {  Task.Run(()=>  {  try  {  var htmlNodes = Parse\_as\_SelectNodes(@"http://procweti.ru/kategorii.php"  , ".//div[@class='wrapper content']" +  "//div[@id='right-col']" +  "//div[@class='js\_shop\_list shop\_list']" +  "//div[@class='shop\_cat\_link']" + |

Продолжение Листинга 3

|  |
| --- |
| "//a"  );  foreach (var item in htmlNodes)  {  Match match = Regex.Match(item.OuterHtml, @".\*?href=""(.\*?)"".\*?");  var htmlPlantCategory = Parse\_as\_SelectNode(match.Groups[1].Value  , ".//div[@class='wrapper content']" +  "//div[@id='right-col']" +  "//h1"  );  Match matchPlantCategory = Regex.Match(htmlPlantCategory.OuterHtml, @"<h1>(.\*?)</h1>");  List<string> linksPagePlant = new List<string>();  linksPagePlant.Add(match.Groups[1].Value);  var htmlPlantPageLinks = Parse\_as\_SelectNodes(match.Groups[1].Value  , ".//div[@class='wrapper content']" +  "//div[@id='right-col']" +  "//div[@class='js\_shop\_list shop\_list']" +  "//div[@class='block paginator']" + "//a"  );  if (htmlPlantPageLinks != null)  {  foreach (var itemLinksPagePlant in htmlPlantPageLinks)  {  Match matchLinksPagePlant = Regex.Match(itemLinksPagePlant.OuterHtml, @".\*?href=""(.\*?)"".\*?");  linksPagePlant.Add(matchLinksPagePlant.Groups[1].Value);  }  }  Console.WriteLine(matchPlantCategory.Groups[1].Value);  foreach (var itemLinksPlantCategory in linksPagePlant)  {  var htmlPlantLinks = Parse\_as\_SelectNodes(itemLinksPlantCategory  , ".//div[@class='wrapper content']" +  "//div[@id='right-col']" +  "//div[@class='js\_shop\_list shop\_list']" +  "//div[@class='shop-pane']" +  "//div[@class='js\_shop shop-item shop']" +  "//a[@class='shop-item-title']");  foreach (var itemPlant in htmlPlantLinks)  { |

Продолжение Листинга 3

|  |
| --- |
| Match matchPlants = Regex.Match(itemPlant.OuterHtml, @".\*?href=""(.\*?)"".\*?");  var htmlNodesPlantName = Parse\_as\_SelectNode(matchPlants.Groups[1].Value  , ".//div[@class='wrapper content']" +  "//div[@id='right-col']" +  "//h1"  );  Match matchPlantName = Regex.Match(htmlNodesPlantName.OuterHtml, @"<h1>(.\*?)</h1>");  //richTextBox1.Text += "\t - " + matchPlantName.Groups[1].Value + "\r\n\r\n";  //Латинское название, Семейство, Другие названия  var htmlNodesPlantINFO = Parse\_as\_SelectNode(matchPlants.Groups[1].Value  , ".//div[@class='wrapper content']" +  "//div[@id='right-col']" +  "//div[@class='js\_shop\_id js\_shop shop shop\_id shop-item-container']" +  "//div[@class='shop\_text']");  //Латинское название  Match matchPlantLatinicName = Regex.Match(htmlNodesPlantINFO.OuterHtml, @".\*?<p><u>Латинское название:</u> (.\*?).</p>.\*?");  //Семейство  Match matchPlantFamily = Regex.Match(htmlNodesPlantINFO.OuterHtml, @".\*?<p><u>Семейство:</u> (.\*?).</p>.\*?");  //Другие названия  Match matchPlantAnotherName = Regex.Match(htmlNodesPlantINFO.OuterHtml, @".\*?<p><u>Другие названия:</u> (.\*?).</p>.\*?");  MatchCollection collectionDescription = Regex.Matches(htmlNodesPlantINFO.OuterHtml, @"<p>(.\*?)</p>");  string description = "";  foreach (Match itemDescription in collectionDescription)  {  string partDesc = itemDescription.Groups[1].Value;  if (partDesc.IndexOf("<u>") == -1)  {  partDesc.Replace("<p>", "\t");  partDesc.Replace("</p>", "");  description += partDesc;  }  }  Invoke((MethodInvoker)delegate  {  var Row = plantDataSet.Plant.NewRow();  Row["Name"] = matchPlantName.Groups[1].Value; |

Окончание Листинга 3

|  |
| --- |
| Row["Category"] = matchPlantCategory.Groups[1].Value;  Row["LatinName"] = matchPlantLatinicName.Groups[1].Value;  Row["Family"] = matchPlantFamily.Groups[1].Value;  Row["AnotherName"] = matchPlantAnotherName.Groups[1].Value;  Row["URL"] = matchPlants.Groups[1].Value;  if (plantDataSet.Plant.Select("Name='" + Row["Name"] + "'").Length == 0)  {  plantDataSet.Plant.Rows.Add(Row);  plantTableAdapter.Update(plantDataSet.Plant);  plantDataSet.AcceptChanges();  plantDataGridView.DataSource = plantBindingSource;  }  });  }  }  }  MessageBox.Show("Окончание Парсинга","Паринг сайта");  Updating = false;  }  catch (Exception ex)  {  Console.WriteLine("What's wrong. {Exception}");  }  }); |

С помощью языка запросов XPath находятся все категории, страницы с растениями из данной категориями и всеми растениями с данных страниц. Затем используется регулярное выражение для получения интересующей информации. 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 Обработчики на главной форме

|  |
| --- |
| private void bindingNavigatorDeleteItem\_Click(object sender, EventArgs e)  {  var rowIndex = this.plantDataGridView.SelectedCells[0].RowIndex;  plantDataGridView.Rows.RemoveAt(rowIndex);  plantDataSet.Tables["Plant"].Rows[rowIndex].Delete();  plantTableAdapter.Update(plantDataSet);  }  private void toolStripButtonAdd\_Click(object sender, EventArgs e)  {  var bank = new PlantForm();  bank.ShowDialog();  this.plantTableAdapter.Fill(this.plantDataSet.Plant);  } |

Продолжение Листинга4

|  |
| --- |
| private void toolStripButtonEdit\_Click(object sender, EventArgs e)  {  if (plantDataGridView.SelectedRows.Count > 0)  {  var plant = new PlantForm(plantDataGridView.SelectedCells[0].RowIndex);  plant.ShowDialog();  this.plantTableAdapter.Fill(this.plantDataSet.Plant);  }  } |

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 Обработчики на второй форме

|  |
| --- |
| public PlantForm(int pos = -1)  {  InitializeComponent();  this.plantTableAdapter.Fill(this.plantDataSet.Plant);  if (pos == -1)  {  plantBindingSource.AddNew();  plantBindingSource.MoveLast();  }  else plantBindingSource.Position = pos;  }  private void plantBindingNavigatorSaveItem\_Click(object sender, EventArgs e)  {  this.Validate();  this.plantBindingSource.EndEdit();  this.tableAdapterManager.UpdateAll(this.plantDataSet);  }  private void buttonSave\_Click(object sender, EventArgs e)  {  try  {  if (string.IsNullOrWhiteSpace(nameTextBox.Text) || string.IsNullOrWhiteSpace(categoryTextBox.Text) || string.IsNullOrWhiteSpace(familyTextBox.Text) || string.IsNullOrWhiteSpace(uRLTextBox.Text))  {  MessageBox.Show("Данные заполнены некорректно!", "Ошибка");  }  else  {  this.Validate();  this.plantBindingSource.EndEdit();  this.tableAdapterManager.UpdateAll(this.plantDataSet);  Close();  } |

Продолжение Листинга 6

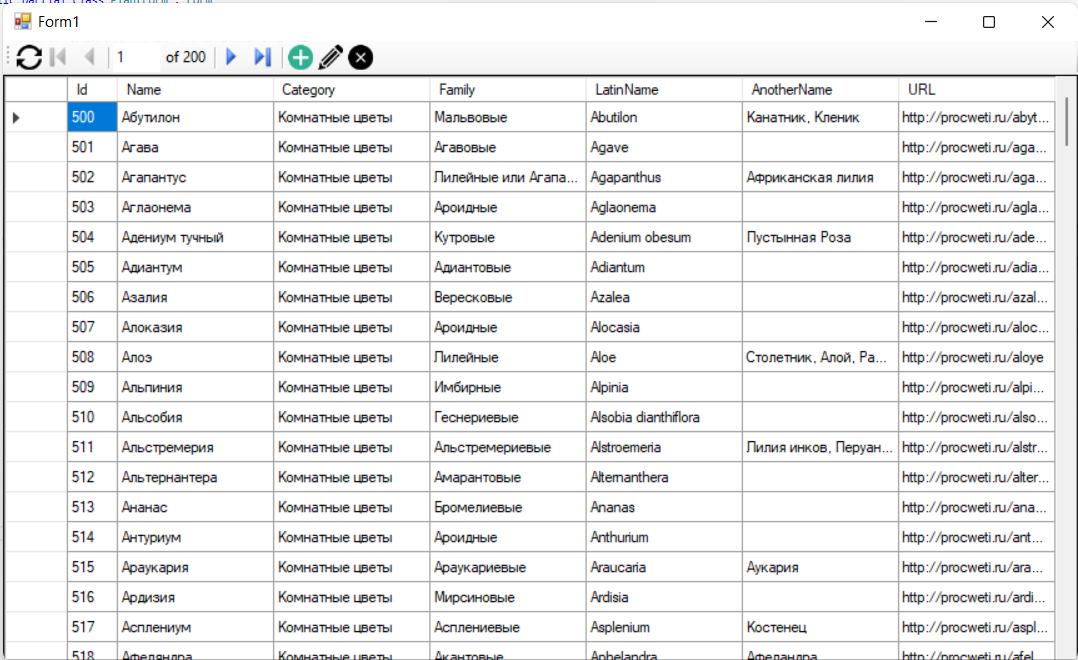
|  |
| --- |
| }  catch (System.Exception exception)  {  MessageBox.Show(exception.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);  }  } |

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

# ОПИСАНИЕ ИНТЕРФЕЙСА РАЗРАБОТАННОЙ ПРОГРАММЫ

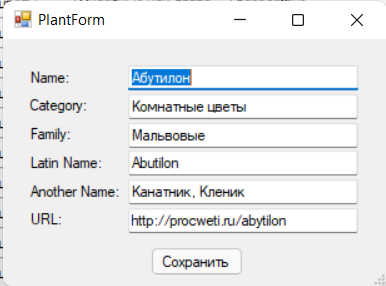
Для создания интерфейса в расчётно-графической работе использовалось приложение Windows Forms, в котором была создана главная форма приложения (рис. 7), включающая в себя представленный список компонентов:

* 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. 7. Интерфейс приложения**

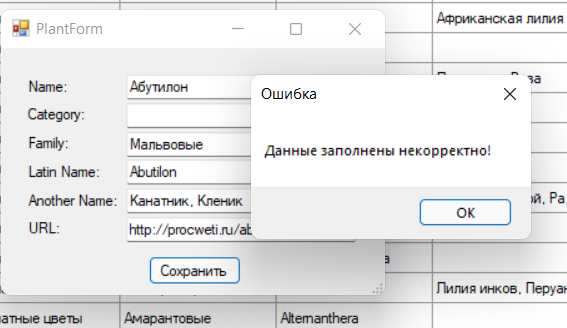
После загрузки данных с веб-сайта приложение принимает вид, показанный на рис. 7.



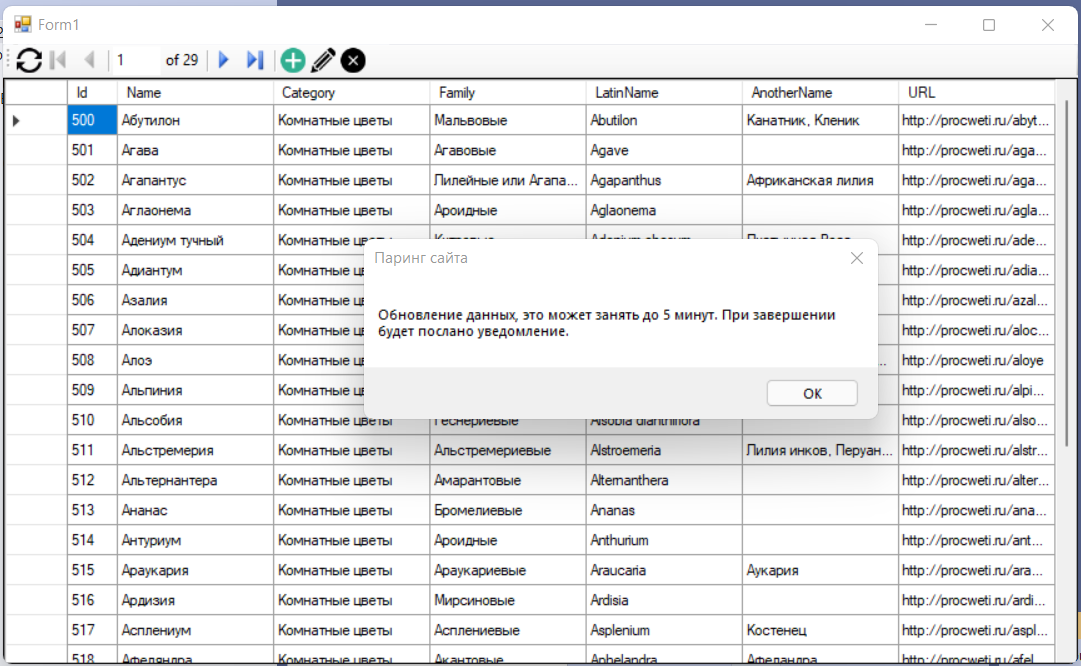
**Рис. 8. Всплывающее окно для изменения**

# ОПИСАНИЕ ПРОЦЕССА ТЕСТИРОВАНИЯ ПРОГРАММЫ

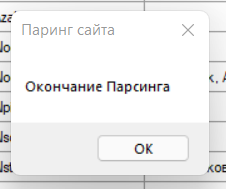
При попытке добавить в таблицу пустое значение приложение обработает эту ситуацию и выдаст сообщение об ошибке.



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



**Рис. 10.** Уведомлениеначалепарсинга сайта



**Рис. 11.** Уведомление об окончании парсинга сайта

# 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).