Отчет по практическим работам № 4, 5, 6

Тема: «Разработать программные модули страниц

CRUD:

- просмотра данных из таблиц БД

- реализовать модули добавления данных, редактирования

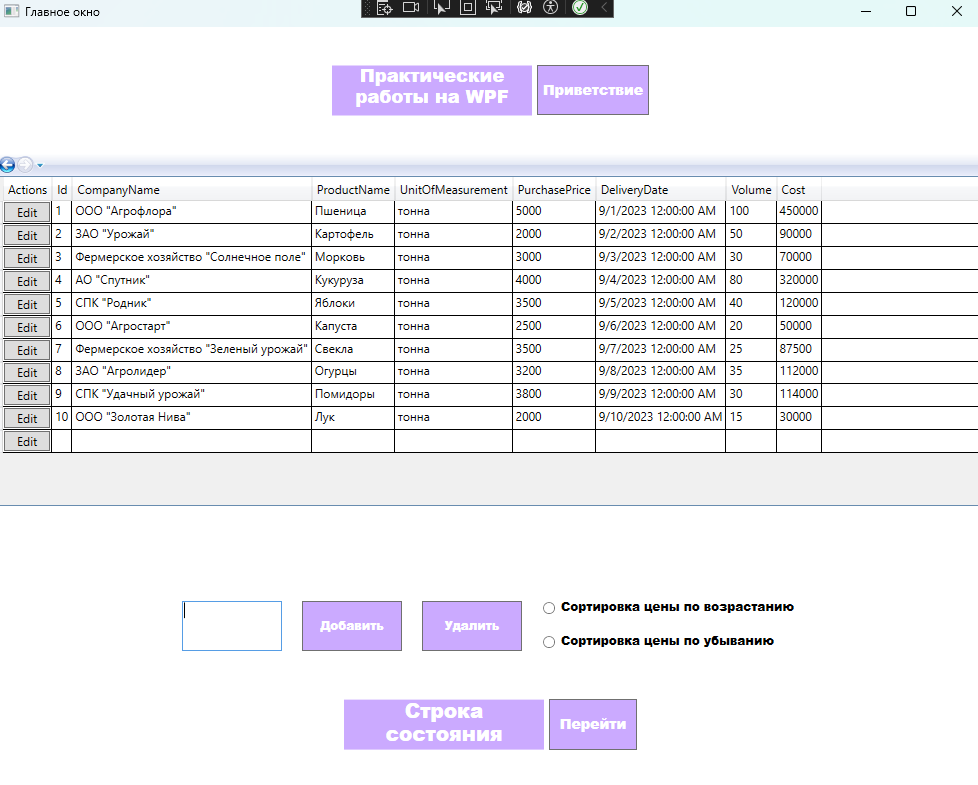
- реализовать модуль удаления данных.»

Выполнил: студент гр. ИСП.21.1А Жигач Дмитрий

Вариант № 9. Предметная область: Сельскохозяйственные работы.  
  
<!-- Я решил объединить эти практические в одну так как тут всё взаимосвязано весь код написан в файле pr8.xaml.cs-->

1. Создал чтение элементов из базы данных и сохранение их в переменные списка в классе DatabaseHelper, теперь в таблице отображаются все строки и столбцы.  
     
   public class DatabaseHelper  
   {  
   private string \_connectionString = "Data Source=F:\\Приложение на WPF\\database\\PR1SQL.db;Version=3;";  
     
   public List<Item> GetItems()  
   {  
   using (SQLiteConnection connection = new SQLiteConnection(\_connectionString))  
   {  
   connection.Open();  
   string query = "SELECT \* FROM products";  
   SQLiteCommand command = new SQLiteCommand(query, connection);  
   SQLiteDataReader reader = command.ExecuteReader();  
     
   List<Item> items = new List<Item>();  
   while (reader.Read())  
   {  
   items.Add(new Item  
   {  
   Id = Convert.ToInt32(reader["id"]),  
   CompanyName = reader["company\_name"].ToString(),  
   ProductName = reader["product\_name"].ToString(),  
   UnitOfMeasurement = reader["unit\_of\_measurement"].ToString(),  
   PurchasePrice = Convert.ToDouble(reader["purchase\_price"]),  
   DeliveryDate = DateTime.Parse(reader["delivery\_date"].ToString()),  
   Volume = Convert.ToDouble(reader["volume"]),  
   Cost = Convert.ToDouble(reader["cost"])  
   });  
   }  
   return items;  
   }  
   }
2. Создал добавление элементов в таблицу с сохранением в бд.  
     
   public void AddItem(Item item)  
   {  
   using (SQLiteConnection connection = new SQLiteConnection(\_connectionString))  
   {  
   connection.Open();  
   string maxIdQuery = "SELECT MAX(id) FROM products";  
   SQLiteCommand maxIdCommand = new SQLiteCommand(maxIdQuery, connection);  
   int maxId = Convert.ToInt32(maxIdCommand.ExecuteScalar());  
   item.Id = maxId + 1  
     
   string query = "INSERT INTO products (id, company\_name, product\_name, unit\_of\_measurement, purchase\_price, delivery\_date, volume, cost) VALUES (@Id, @CompanyName, @ProductName, @UnitOfMeasurement, @PurchasePrice, @DeliveryDate, @Volume, @Cost)";  
   SQLiteCommand command = new SQLiteCommand(query, connection);  
   command.Parameters.AddWithValue("@Id", item.Id);  
   command.Parameters.AddWithValue("@CompanyName", item.CompanyName);  
   command.Parameters.AddWithValue("@ProductName", item.ProductName);  
   command.Parameters.AddWithValue("@UnitOfMeasurement", item.UnitOfMeasurement);  
   command.Parameters.AddWithValue("@PurchasePrice", item.PurchasePrice);  
   command.Parameters.AddWithValue("@DeliveryDate", item.DeliveryDate);  
   command.Parameters.AddWithValue("@Volume", item.Volume);  
   command.Parameters.AddWithValue("@Cost", item.Cost);  
   command.ExecuteNonQuery();  
   }  
   }
3. Создал удаление элементов из таблицы с сохранением в бд.  
     
   public void DeleteItem(int itemId)  
   {  
   using (SQLiteConnection connection = new SQLiteConnection(\_connectionString))  
   {  
   connection.Open();  
   string query = "DELETE FROM products WHERE id = @Id";  
   SQLiteCommand command = new SQLiteCommand(query, connection);  
   command.Parameters.AddWithValue("@Id", itemId);  
   command.ExecuteNonQuery();  
   }  
   }
4. Создал редактирование элементов из таблицы.  
     
   public void UpdateItem(Item updatedItem)  
   {  
   using (SQLiteConnection connection = new SQLiteConnection(\_connectionString))  
   {  
   connection.Open();  
   string query = "UPDATE products SET company\_name = @CompanyName, product\_name = @ProductName, unit\_of\_measurement = @UnitOfMeasurement, purchase\_price = @PurchasePrice, delivery\_date = @DeliveryDate, volume = @Volume, cost = @Cost WHERE id = @Id";  
   SQLiteCommand command = new SQLiteCommand(query, connection);  
   command.Parameters.AddWithValue("@Id", updatedItem.Id);  
   command.Parameters.AddWithValue("@CompanyName", updatedItem.CompanyName);  
   command.Parameters.AddWithValue("@ProductName", updatedItem.ProductName);  
   command.Parameters.AddWithValue("@UnitOfMeasurement", updatedItem.UnitOfMeasurement);  
   command.Parameters.AddWithValue("@PurchasePrice", updatedItem.PurchasePrice);  
   command.Parameters.AddWithValue("@DeliveryDate", updatedItem.DeliveryDate);  
   command.Parameters.AddWithValue("@Volume", updatedItem.Volume);  
   command.Parameters.AddWithValue("@Cost", updatedItem.Cost);  
   command.ExecuteNonQuery();  
   }  
   }  
   }
5. Создал кнопку для добавления элементов в таблицу с заданными начальными параметрами.  
     
   private void AddButton\_Click(object sender, RoutedEventArgs e)  
   {  
   Item newItem = new Item  
   {  
   CompanyName = "Новая компания",  
   ProductName = "Новый продукт",  
   UnitOfMeasurement = "шт.",  
   PurchasePrice = 0.0,  
   DeliveryDate = DateTime.Now,  
   Volume = 0.0,  
   Cost = 0.0  
   };  
   \_dbHelper.AddItem(newItem);  
   \_items = \_dbHelper.GetItems();  
   dataGrid.ItemsSource = \_items;  
   }
6. Создал кнопку для удаления элементов из таблицы.  
     
   private void DeleteButton\_Click(object sender, RoutedEventArgs e)  
   {  
   if (dataGrid.SelectedItem != null)  
   {  
   Item selectedItem = (Item)dataGrid.SelectedItem;  
   int itemId = selectedItem.Id;  
   \_dbHelper.DeleteItem(itemId);  
   \_items = \_dbHelper.GetItems();  
   dataGrid.ItemsSource = \_items;  
   }  
   else  
   {  
   MessageBox.Show("Выберите строку для удаления.");  
   }  
   }
7. Вспомнил про фильтрацию для таблицы, создал фильтр для неё.  
     
   private void SearchTextBox\_TextChanged(object sender, TextChangedEventArgs e)  
   {  
   string filterText = SearchTextBox.Text.ToLower();  
   List<Item> filteredItems = \_items.Where(item =>  
   item.CompanyName.ToLower().Contains(filterText) ||  
   item.ProductName.ToLower().Contains(filterText) ||  
   item.UnitOfMeasurement.ToLower().Contains(filterText) ||  
   item.PurchasePrice.ToString().Contains(filterText) ||  
   item.DeliveryDate.ToString().Contains(filterText) ||  
   item.Volume.ToString().Contains(filterText) ||  
   item.Cost.ToString().Contains(filterText)  
   ).ToList();  
   dataGrid.ItemsSource = filteredItems;  
   }
8. Создал сортировку по возрастанию и убыванию цены покупки.  
     
   private void RbtnUp\_click(object sender, RoutedEventArgs e)  
   {  
   \_items = \_items.OrderBy(item => item.PurchasePrice).ToList();  
   dataGrid.ItemsSource = \_items;  
   }  
     
   private void RbtnDown\_click(object sender, RoutedEventArgs e)  
   {  
   \_items = \_items.OrderByDescending(item => item.PurchasePrice).ToList();  
   dataGrid.ItemsSource = \_items;  
   }
9. Создал кнопку редактирования.  
     
   private void EditButton\_Click(object sender, RoutedEventArgs e)  
   {  
   DataGridCellInfo cellInfo = dataGrid.CurrentCell;  
   if (cellInfo.IsValid && cellInfo.Column != null)  
   {  
   DataGridCell cell = new DataGridCell  
   {  
   Content = cellInfo.Item,  
   Style = cellInfo.Column.CellStyle  
   };  
     
   if (cell.IsEditing)  
   {  
   Item editedItem = (Item)cellInfo.Item;  
   \_dbHelper.UpdateItem(editedItem);  
   }  
   }  
   }  
     
   Вот логика сохранения после редактирования ячейки.  
     
   private void DataGrid\_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e)  
   {  
   if (e.EditAction == DataGridEditAction.Commit)  
   {  
   string header = e.Column.Header.ToString();  
   string newValue = (e.EditingElement as TextBox)?.Text;  
   Item editedItem = e.Row.Item as Item;  
     
   switch (header)  
   {  
   case "Id": // Id  
   int id = int.Parse(newValue);  
   if (editedItem.Id != id)  
   {  
   MessageBox.Show("Id should not be edited!");  
   return;  
   }  
   break;  
   case "CompanyName": // CompanyName  
   if (editedItem.CompanyName != newValue)  
   {  
   editedItem.ProductName = newValue;  
   }  
   break;  
   case "UnitOfMeasurement": // UnitOfMeasurement  
   if (editedItem.UnitOfMeasurement != newValue)  
   {  
   editedItem.UnitOfMeasurement = newValue;  
   }  
   break;  
   case "PurchasePrice": // PurchasePrice  
   double purchasePrice = double.Parse(newValue);  
   if (editedItem.PurchasePrice != purchasePrice)  
   {  
   editedItem.PurchasePrice = purchasePrice;  
   }  
   break;  
   case "DeliveryDate": // DeliveryDate  
   DateTime deliveryDate = DateTime.Parse(newValue);  
   if (editedItem.DeliveryDate != deliveryDate)  
   {  
   editedItem.DeliveryDate = deliveryDate;  
   }  
   break;  
   case "Volume": // Volume  
   double volume = double.Parse(newValue);  
   if (editedItem.Volume != volume)  
   {  
   editedItem.Volume = volume;  
   }  
   break;  
   case "Cost": // Cost  
   double cost = double.Parse(newValue);  
   if (editedItem.Cost != cost)  
   {  
   editedItem.Cost = cost;  
   }  
   break;  
   }  
   \_dbHelper.UpdateItem(editedItem);  
   }  
   }  
   }  
   }

Итоговый вид приложения



А вот разметка pr8.xaml

<Grid>

<DataGrid x:Name="dataGrid"

IsReadOnly="False"

CellEditEnding="DataGrid\_CellEditEnding"

AutoGenerateColumns="True"

HorizontalAlignment="Left"

VerticalAlignment="Top"

Height="330"

Width="1000">

<DataGrid.Columns>

<DataGridTemplateColumn Header="Actions">

<DataGridTemplateColumn.CellTemplate>

<DataTemplate>

<Button Content="Edit" Click="EditButton\_Click"/>

</DataTemplate>

</DataGridTemplateColumn.CellTemplate>

</DataGridTemplateColumn>

</DataGrid.Columns>

</DataGrid>

<DockPanel VerticalAlignment="Bottom"

HorizontalAlignment="Center">

<TextBox x:Name="SearchTextBox"

Width="100"

Height="50"

Margin="10"

TextChanged="SearchTextBox\_TextChanged"/>

<Button Content="Добавить"

HorizontalAlignment="Left"

VerticalAlignment="Bottom"

Width="100"

Background="#FFCBAAFF" Foreground="White" FontFamily="Arial Black"

Click="AddButton\_Click" Height="50"

Margin="10"/>

<Button Content="Удалить"

HorizontalAlignment="Left"

VerticalAlignment="Bottom"

Width="100"

Background="#FFCBAAFF" Foreground="White" FontFamily="Arial Black"

Click="DeleteButton\_Click" Height="50"

Margin="10"/>

<StackPanel>

<RadioButton

Margin="10"

Name="RbtUp"

Content="Сортировка цены по возрастанию"

FontFamily="Arial Black"

Click="RbtnUp\_click"/>

<RadioButton

Margin="10"

Name="RbtnDown"

Content="Сортировка цены по убыванию"

FontFamily="Arial Black"

Click="RbtnDown\_click"/>

</StackPanel>

</DockPanel>

</Grid>