Xamarin

SetContentView(Resource.Layout.Main); //присоединить layout к активности

Button button = FindViewById<Button>(Resource.Id.myButton);

button.Click += Button\_order; //повесить событие на кнопку

StartActivity(typeof(ActivityAbout)); //открыть активность

WPF

|  |
| --- |
| public static class Entity // класс для entity  {  private static EntitiesSew entities = new EntitiesSew();  private static Пользователь user = null;  public static EntitiesSew Entities { get => entities; }  public static Пользователь User { get => user; set => user = value; }  } |
| public static class Windows // класс для открытия новой формы взамен старой  {  public static void ShowWindow(dynamic thisForm, dynamic nextForm)  {  thisForm.Hide();  nextForm.Show();  thisForm.Close();  }  } |
| var user = Entity.Entities.Пользователь.Where(u => u.Логин == TLogin.Text && u.Пароль == TPassword.Text).First(); //выбор пользователя из базы  if (user != null) //открытие формы вошедшего пользователя  {  Entity.User = user;  switch (Entity.User.Роль)  {  case "Заказчик":  Windows.ShowWindow(this, new ClientWindow());  break;  case "Кладовщик":  MessageBox.Show("");  break;  case "Менеджер":  Windows.ShowWindow(this, new ManagerWindow());  break;  case "Руководитель":  MessageBox.Show("");  break;  default:  break;  }  } |
| Пользователь пользователь = new Пользователь()  {  Логин = TLogin.Text,  Пароль = TPassword.Text  };  Entity.Entities.Пользователь.Add(пользователь);  Entity.Entities.SaveChanges(); |
| grid.ItemsSource = Entity.Entities.Table.ToList(); //заполнение списка данными |
| dynamic row = grid.SelectedItem; //запись строки |
| grid.Columns[3].Visibility = Visibility.Collapsed; //скрыть некоторую строку |
| if (item == 0)  {  Entity.Entities.Table.Add(new Table() // добавление нового объекта  {  name = name.Text,  cost = Convert.ToInt32(cost.Text),  image = buffer  });  }  Else //изменение существующего объекта  {  Table table = Entity.Entities.Table.Where(i => i.ID == item).FirstOrDefault();  table.name = name.Text;  table.cost = Convert.ToInt32(cost.Text);  table.image = buffer;  }  Entity.Entities.SaveChanges();//сохранение изменений !!!!!!!!! |
| Запись картинки в массив байт  OpenFileDialog fileDialog = new OpenFileDialog();//диалоговое окно открытия файла  if(fileDialog.ShowDialog()==true)  {  FileStream fileStream = new FileStream(fileDialog.FileName, FileMode.Open, FileAccess.Read);//открытие потока данных  buffer = new byte[fileStream.Length]; //инициализация буфера  fileStream.Read(buffer, 0, (int)(fileStream.Length));  fileStream.Close();  image.Source = GetBitmapSource(buffer);  } |
| private static BitmapSource GetBitmapSource(byte[] buf)//формирование картинки из буфера  {  if (buf != null)  {  BitmapImage bitmap = new BitmapImage();  var stream = new MemoryStream(buf);  bitmap.BeginInit();  bitmap.StreamSource = stream;  bitmap.EndInit();  return bitmap as BitmapSource;  }  else  {  return null;  }  } |
| Try // заполнениe поля картинка  {  image.Source = GetBitmapSource(table.image);  }  catch (Exception)  {  image.Source = null;  } |
| Конструктор |
| string filename="";  Microsoft.Win32.OpenFileDialog dialog = new Microsoft.Win32.OpenFileDialog;  if (dialog.ShowDialog() == true)  {  filename = dialog.FileName;  }  Rectangle rectangle = new Rectangle()  {  Width = 150,  Height = 150,  Margin = new Thickness(10, 10, 0, 0),  Uid = "rec" + (++Reci),  Fill = new ImageBrush(new BitmapImage(new Uri(filename))),  VerticalAlignment = VerticalAlignment.Top,  HorizontalAlignment = HorizontalAlignment.Left  };    rectangle.MouseDown += Rec\_Down;  rectangle.MouseUp += Rec\_Up;  rectangle.MouseMove += Rec\_MouseMove;  canvas.Children.Add(rectangle); |
| private void Rec\_Down(object sender, RoutedEventArgs e)  {  try  {  thisRec.Stroke = null;  }  catch (Exception)  {  }  finally  {  thisRec = (Rectangle)(sender);  thisRec.Stroke = new SolidColorBrush(Colors.Black);  drop = true;  }  } |
| private void Rec\_Up(object sender, RoutedEventArgs e)  {  drop = false;  } |
| private void Rec\_MouseMove(object sender, MouseEventArgs e) //выбор точки перемещения прямоугольника, если зажата кнопка мыши  {  if (drop)  {  double centrex = thisRec.Width / 2;  double centrey = thisRec.Height / 2;  double cx = e.GetPosition(canvas).X;  double cy = e.GetPosition(canvas).Y;    thisRec.Margin = new Thickness((cx- centrex), (cy-centrey), 0, 0);  }  } |
| private void Button\_Click\_1(object sender, RoutedEventArgs e)  {  Microsoft.Win32.SaveFileDialog dialog = new Microsoft.Win32.SaveFileDialog() { FileName = "Без названия.png" };  if (dialog.ShowDialog() == true)  {  RenderTargetBitmap render = new RenderTargetBitmap((int)(canvas.Width), (int)(canvas.Height), 96, 96, PixelFormats.Pbgra32);  render.Render(canvas);  using (var file = System.IO.File.Open(dialog.FileName, System.IO.FileMode.OpenOrCreate))  {  PngBitmapEncoder encoder = new PngBitmapEncoder();  encoder.Frames.Add(BitmapFrame.Create(render));  encoder.Save(file);  }  }  } |
| private void Window\_KeyUp(object sender, KeyEventArgs e)  {  resize = false;  } |
| private void Window\_KeyDown(object sender, KeyEventArgs e)  {  if (e.Key == Key.LeftCtrl || e.Key == Key.RightCtrl)  {  resize = true;  }  } |
| private void canvas\_MouseWheel(object sender, MouseWheelEventArgs e)  {  try  {  int delta = (e.Delta / 12); // градус поворота колеса мыши  if (resize)  {//изменение размеров  thisRec.Width = thisRec.Width += delta;  thisRec.Height = thisRec.Height += delta;  }  else  { //получение угла поворота  RotateTransform rotate = thisRec.RenderTransform as RotateTransform;  //поворот прямоугольника на заданный угол  try  {  thisRec.RenderTransform = new RotateTransform((int)(rotate.Angle += delta), thisRec.Width / 2, thisRec.Height / 2);  }  catch (Exception)  {  thisRec.RenderTransform = new RotateTransform(delta, thisRec.Width / 2, thisRec.Height / 2);  }  }  }  catch (Exception)  {  } |
| try  {  var fabric = from p in Entity.Entities.Изделия\_  join pf in Entity.Entities.ТканиИзделия on p.Артикул equals pf.Изделиe  join f in Entity.Entities.СкладТкани on pf.Ткань equals f.Ткань  where p.Артикул == thisitem.ToString()  select new  {  x = (double)f.Ширина,  y = (double)f.Длина  }; //извлечение длины и ширины рулона ткани    //задание размеров полотна  canvas.Height = fabric.FirstOrDefault().x;  canvas.Width = fabric.FirstOrDefault().y;  LPrice.Content = canvas.Height + " на " + canvas.Width; //вывод размера полотна    SizeToContent = SizeToContent.WidthAndHeight; //перерисовка окна под размеры содержимого  //стартвая позиция в ноль по canvas  minX = 0;  minY = 0;  //текущая точка тоже в ноль  thisX = minX;  thisY = minY;  // предельная точка  maxY = canvas.Height;  maxX = canvas.Width;  bool newcolumnflag = true; //флаг на перенос на новую колонку  int newcolumncount = 0;  //получение размеров изделий и их количество  var product = from p in Entity.Entities.Изделия\_  join po in Entity.Entities.ЗаказанныеИзделия on p.Артикул equals po.Изделие  where p.Артикул == thisitem.ToString() && po.Заказ.ToString() == order  orderby p.Ширина, p.Длина  select new  {  x = p.Ширина,  y = p.Длина,  count = po.Количество  };  var item = product.FirstOrDefault();  for (int i = 0; i < item.count; i++)  {  // распределение наибольшего У, и наименьшего X  double y = Convert.ToDouble(item.x) > Convert.ToDouble(item.y) ? Convert.ToDouble(item.x) : Convert.ToDouble(item.y);  double x = Convert.ToDouble(item.x) < Convert.ToDouble(item.y) ? Convert.ToDouble(item.x) : Convert.ToDouble(item.y);  //условие на новую колонку, если новый элемент не входит в столбец по Y, то перенос на новый  if (thisY + y > maxY && (thisX + x) <= maxX)  {  thisY = minY;  thisX += x;  newcolumnflag = false;  }  //условие на максимум изделий  if (thisY + y > maxY || (thisX + x) > maxX) return;  //новый прямоугольник с заданными размерами и отступом по текущей точке (х,у)  Rectangle rectangle = new Rectangle()  {  HorizontalAlignment = HorizontalAlignment.Left,  VerticalAlignment = VerticalAlignment.Top,  Uid = "rec" + i.ToString(),  Fill = new SolidColorBrush(Color.FromRgb(224, 169, 175)),  Stroke = new SolidColorBrush(Colors.Black),  Margin = new Thickness(thisX, thisY, 0, 0),  Height = y,  Width = x  };  canvas.Children.Add(element: rectangle);  //условие на отступ в колонке, если помещается - помещаем, нет - выходим.  if (thisY + y <= maxY && thisX + x <= maxX )  {  thisY += y;  if (newcolumnflag) { newcolumncount++; }    }  else  {  return;  }  LPrice.Content = "Размер раскроя: " + (thisX + x) + "x" + (thisY) + ". Списано ткани: " + (maxY-newcolumncount\*y) + "x" + (thisX + x) + ". Доступно для раскроя: " + (i + 1) + " изделий.";  }    }  catch (Exception exp)  {  MessageBox.Show(exp.Message);  }  } |
| private void Button\_Click(object sender, RoutedEventArgs e)  {  Microsoft.Win32.SaveFileDialog saveFile = new Microsoft.Win32.SaveFileDialog()  {  FileName = "Без названия.png"  }; //открытие диалогового окна для сохранения файла  if (saveFile.ShowDialog() == true)  {  RenderTargetBitmap render = new RenderTargetBitmap(pixelWidth: (int)(canvas.RenderSize.Width), pixelHeight: (int)(canvas.RenderSize.Width), dpiX: 96d, dpiY: 96d, pixelFormat: PixelFormats.Pbgra32); //указание размера, разрешения и формата для рендера картинки  render.Render(canvas); //рендер полотна  using (var filename = System.IO.File.Open(saveFile.FileName, System.IO.FileMode.OpenOrCreate))  {  PngBitmapEncoder encoder = new PngBitmapEncoder();  encoder.Frames.Add(BitmapFrame.Create(render));  encoder.Save(filename);  } //открытие файлового потока для записи карты раскроя в выбранный файл  }  } |
| using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"C:\Users\soft\Desktop\1.csv"))  {  foreach (var item in Entity.Entities.Table.ToList())  {  file.WriteLine(item.ID + ";" + item.name + ";" + item.cost + ";");  }  } //открытие и запись в файл данных из списка для печати |
|  |
|  |
|  |