****

**КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО**

**ОБРАЗОВАНИЯ**

|  |
| --- |
|  |
|  |
|  |
|  |

**Отчет**

**Специальность 09.02.02 «Компьютерные сети»**

**Основы объектно-ориентированного программирования**

Выполнила студентка группы 32КС-20

Мусаткина А.С.

Проверила

Кукшева Б.А.

**Москва 2022**

**Оглавление**

[**Введение** 3](#_Toc122977129)

[**1.** **Создание проекта** 4](#_Toc122977130)

[**2.** **Создание главного экрана системы** 4](#_Toc122977131)

[**2.1.** **Дизайн** 4](#_Toc122977132)

[**2.2.** **Создание таймера** 7](#_Toc122977133)

[**3.** **Подготовка для переходов на другие формы** 9](#_Toc122977134)

[**4.** **Подключение базы данных** 10](#_Toc122977135)

[**5.** **Создание страницы спонсора бегуна** 11](#_Toc122977136)

[**5.1.** **Дизайн** 11](#_Toc122977137)

[**5.2.** **Переход на другую форму** 13](#_Toc122977138)

[**5.3.** **Загрузка бегунов в ComboBox** 13](#_Toc122977139)

[**5.4.** **Реализация суммы пожертвования** 15](#_Toc122977140)

[**5.5.** **Прогрузка фонда в зависимости от выбора бегуна** 16](#_Toc122977141)

[**5.6.** **Проверка ввода информации и добавление ее в базу данных** 17](#_Toc122977142)

[**6.** **Форма для подтверждения оплаты** 19](#_Toc122977143)

[**6.1.** **Дизайн** 19](#_Toc122977144)

[**Заключение** 21](#_Toc122977145)

[**Программные коды проекта.** 22](#_Toc122977146)

[**Список источников** 31](#_Toc122977147)

# **Введение**

В данной работе передо мной была поставлена цель *создать приложение по трем видео на языке C# в программе Visual Studio.*

Для достижения цели были поставлены следующие задачи:

* Изучить структуру приложения;
* Научиться создавать приложения на языке С#;
* При помощи видеоматериала изучить команды.

**Подробнее о приложении.**

Данное приложение называется Marathon Skills 2017. Как следует из названия, это приложение для мероприятия Marathon Skills 2017. В нем требуется сделать простое управление, базу данных и доступную информацию для любого пользователя. Помимо этого в этом приложении требуется создать регистрацию для спонсоров.

# **Создание проекта**

В Visual Studio создаём новый проект - выбираем C#, windows, FPF application. В качестве названия ввести MarathonSkills.

# **Создание главного экрана системы**

После создания проекта будем работать с меню приложения, как указано в презентации. Для начала начнём с дизайна.

# **Дизайн**

Добавить RichTextBox, расположить сверху. Добавить Rectangle, расположить снизу. Добавить Button, затем расположить справа снизу.

Затем в правой панели нажать пкм по MarathonSkills, добавить новую папку с названием *«Resource»*, в которой будут храниться все ресурсы для проекта. Нажав пкм по этой папке, добавить словарь с названием Dictionary. В этом словаре будут прописаны стили для элементов главного экрана.

Первым делом, здесь будет прописана информация о главных цветах нашего проекта — серый, светло-серый и темно-серый.

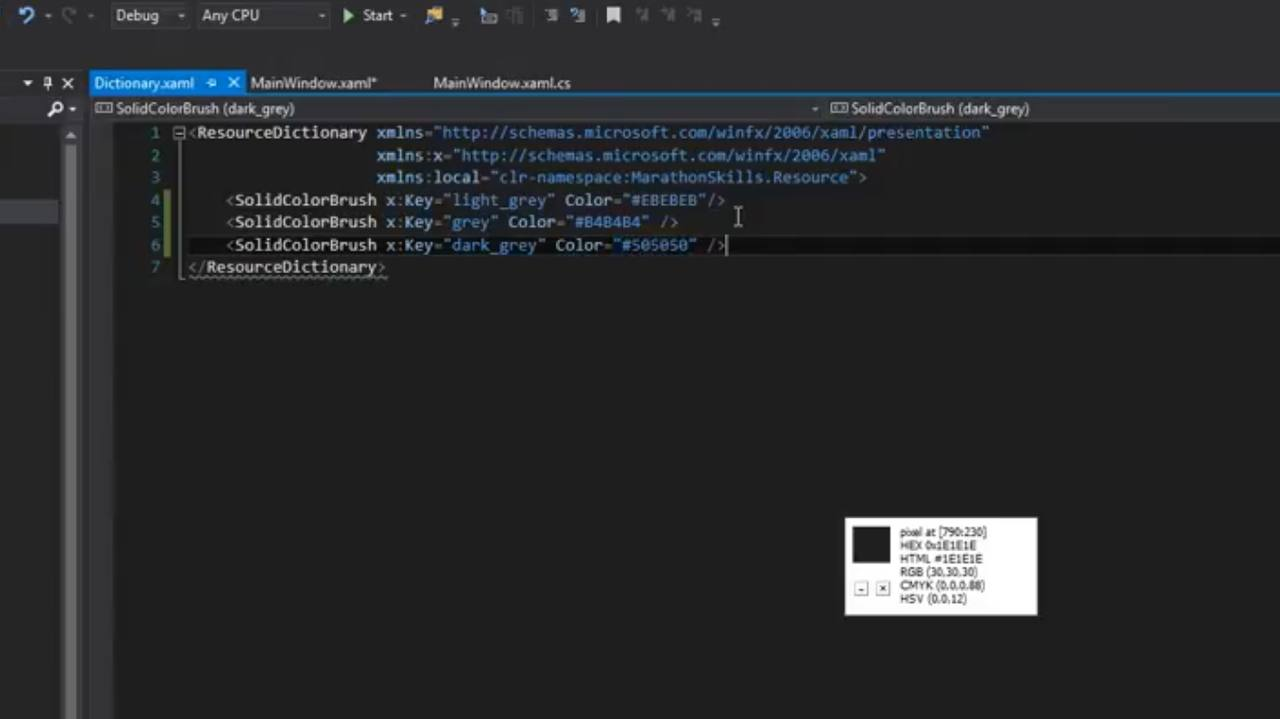


Рисунок 1. Базовые цвета для проекта.

В App.xalm добавим ресурсы приложения, а конкретнее добавим словарь, созданный в прошлом шаге.



Рисунок 2. Словарь связан с проектом.

Возвращаясь на главное окно, в коде, связанный с RichTextBox, можно добавить цвет при помощи команды *Background=“{DynamicResource dark\_grey}”*. Для rectangle пропишем цвет через Fill — *Fill=“{DynamicResource dark\_grey}”.*

В словаре в словаре пропишем стиль для кнопки: добавим цвет кнопки, цвет текста, размер текста, шрифт, размер. Сразу применим его к кнопке через Style — *Style=“{DynamicResource base\_button}”.* Также для этой кнопки меняем имя на button\_becomeRunner.

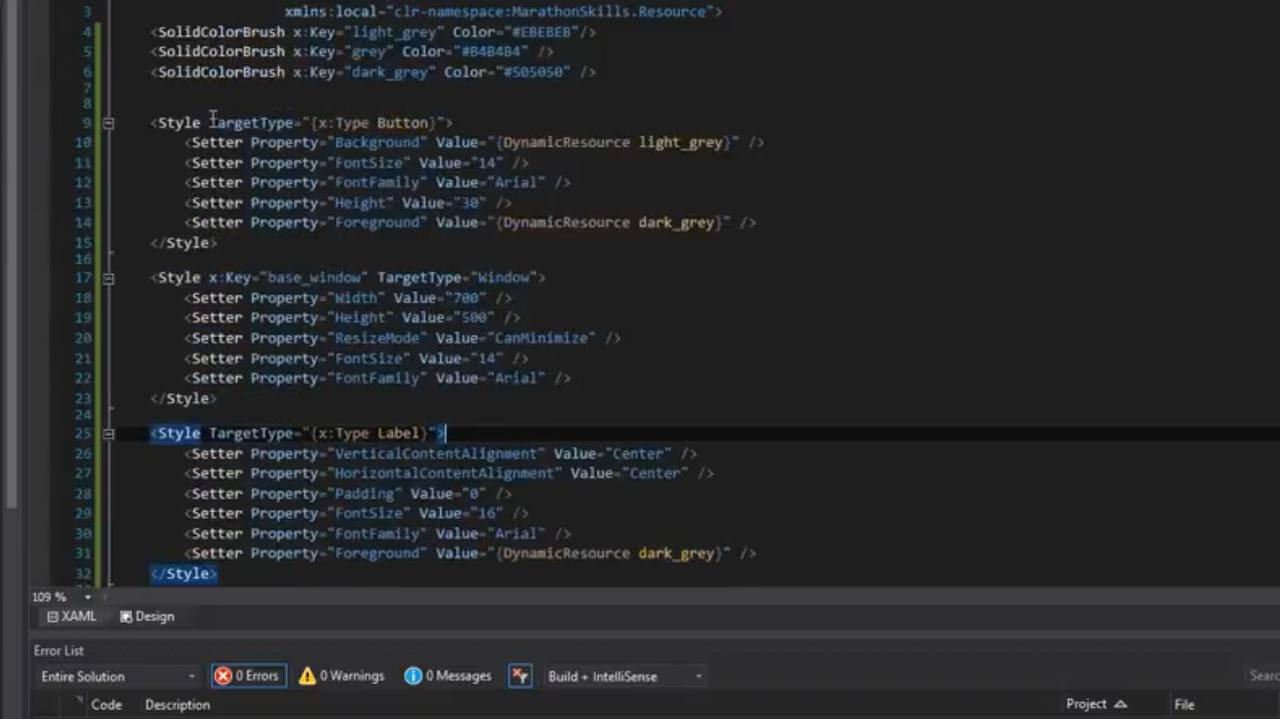


Рисунок 3. Стиль для кнопок.

В свойствах RichTextBox прописать следующее: *BorderThickness=“0” IsReadOnly=“True” Padding=“20” IsHitTestVisible=“False”.* Затем ниже, где прописан Run Text, прописать текст *«MARATHON SKILLS 2017».* Добавить новый параграф, в котором у нас будет прописан следующий текст: «*Москва, Россия»*, переносим текст на новую строку через команду *LineBreak*, и пишем новый текст *«среда 21 октября 2017»*. В свойствах FlowDocument прописываем *TextAligment=“Center”,* который размещает весь текст по центру. Затем прописать свойство *Foreground=“White”*. Данное свойство делает цвет текста в RichTextBox белым. Также пропишем свойство размера текста *FontSize=“20”.* В свойствах первого параграфа прописать свойство *FontWeight=“Bold”,* что делает текст жирным. В свойствах второго параграфа прописать *FontStyle=“Italic”,* который выделяет текст курсивом. Также, в этом параграфе, пропишем цвет для текста *Foreground=“{DynamicResource light\_grey}”.*

Добавим стиль для окна. В нем пропишем ширину, высоту, размер текста, шрифт, а также возможность только свернуть окно. Сразу применим этот стиль к окну. В Tittle прописать *“Style=“{DynamicResource base\_window}”*.

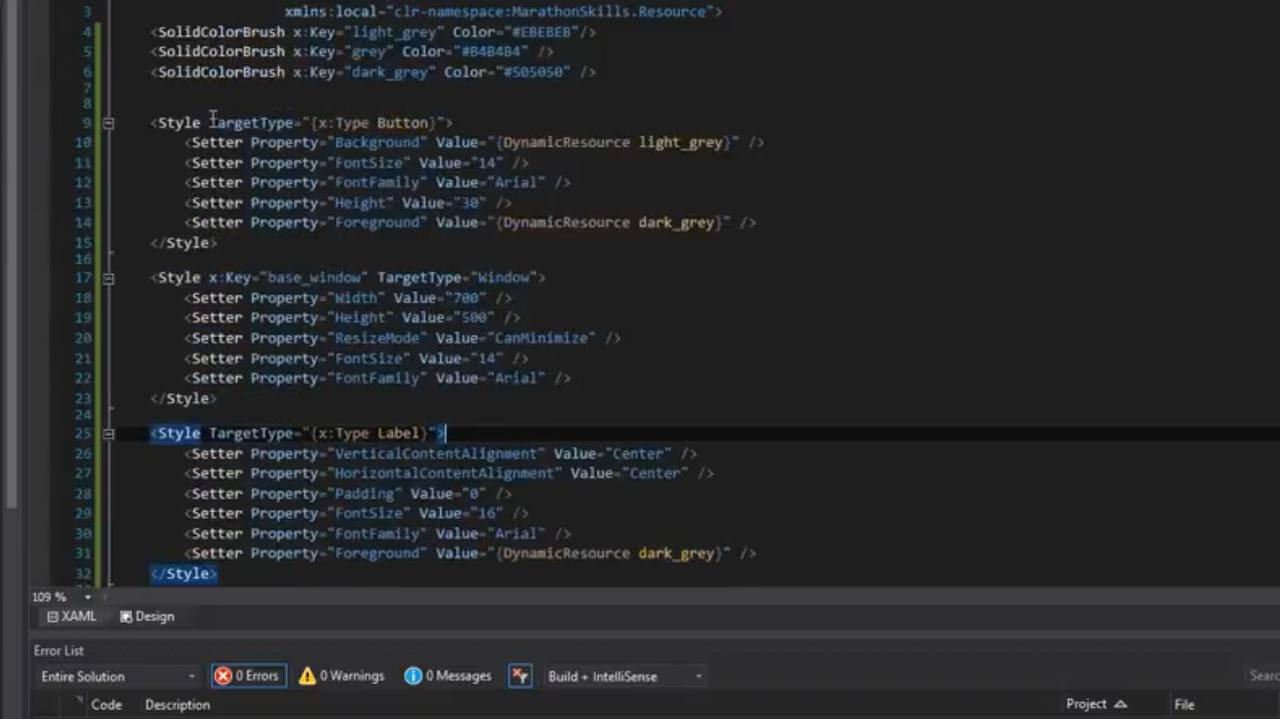


Рисунок 4. Стиль для окна.

В свойствах кнопки напишем фразу «Я хочу стать бегуном». Чтобы текст полностью вмещался, растянуть кнопку до нужного размера, а затем расположить по центру между RichTextBox и Rectangle. Затем скопируем эту кнопку и вставляем три раза. Две кнопки будут находиться над первой, а последняя в правом нижнем углу.

Для кнопки, которая находится ниже, меняем имя на button\_login, прописываем текст Login.

Для второй кнопки меняем имя на button\_becomeSponsor, прописываем текст «Я хочу стать спонсором».

Для третьей кнопки меняем имя на button\_about, прописываем текст «Я хочу узнать больше о событии»

Все три кнопки растягиваем до одинаковых размеров, чтобы в них вмещался текст, и располагаем их по центру друг под другом.

Последним что остаётся — это добавить Label. Его размещаем поверх Rectangle. Задаём ему имя label\_tamer. В контенте оставляем пустым.

Для label создадим стиль. В нем пропишем расположение текста по центру, отступ 0, размер текста 16, тип шрифта Arial, цвет темно-серый.

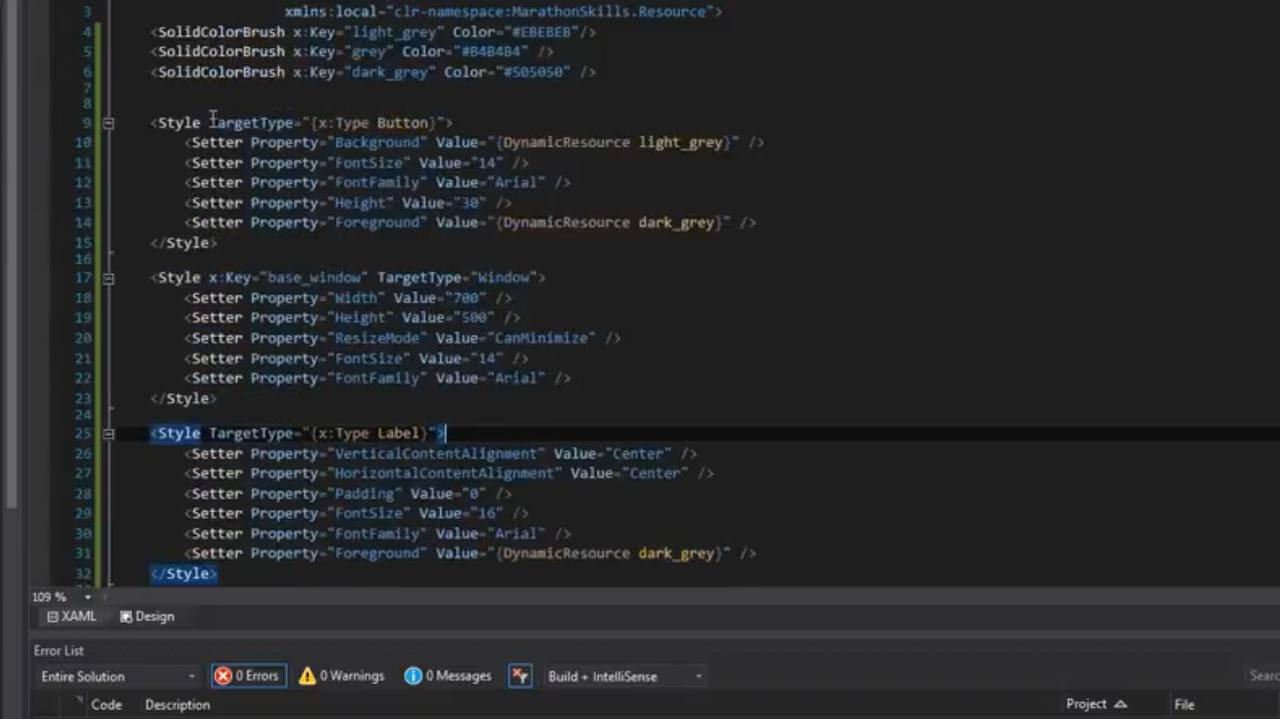


Рисунок 5. Стиль для label.

# **Создание таймера**

В MainWindow.xalm.cs пропишем функцию, которая автоматически будет обновлять наш label и показывать, сколько времени осталось до события.

Добавляем свойство Time типа string. В нем прописываем операцию get, в которой есть первая дата - сейчас, вторая дата начало марафона. Затем идёт расчёт разницы и формирует в строку по формату.

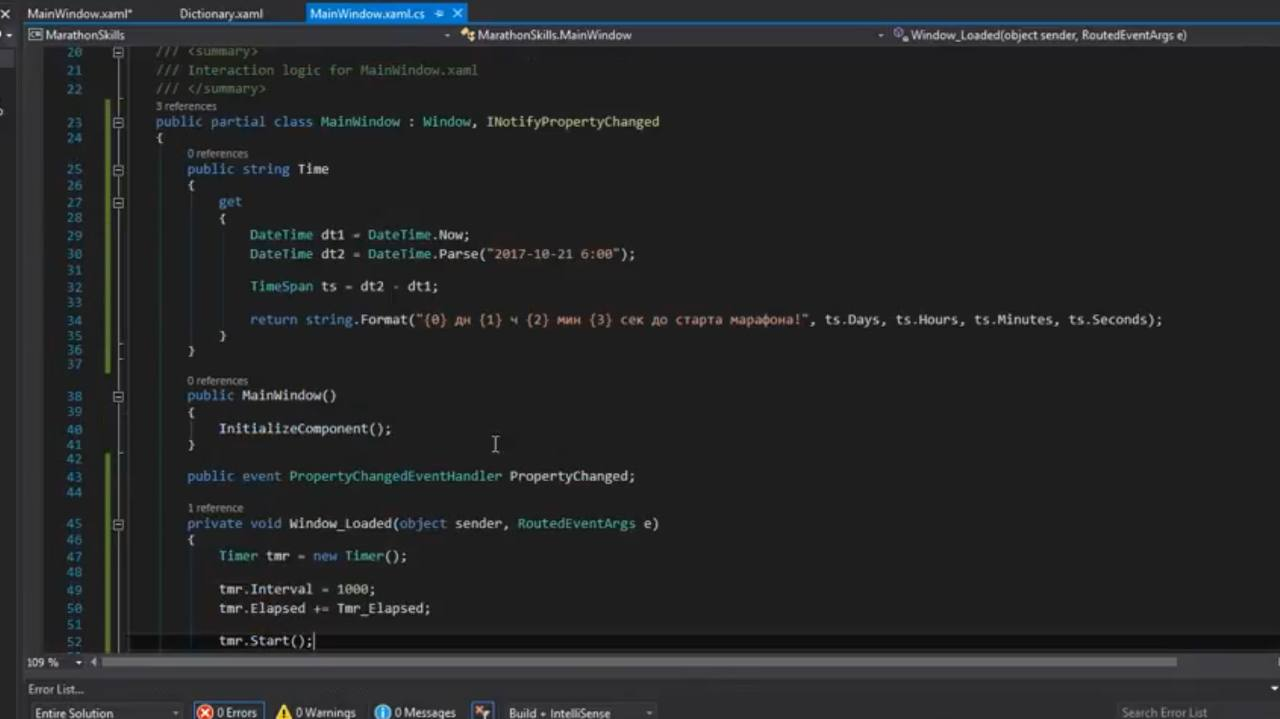


Рисунок 6. Функция Time и Window\_Loaded.

При запуске формы в конструкторе присваиваем текущий контекст.

Также подключен интерфейс IfNotifyPropertyChanged, который вызывает при тиках таймера и сообщает, что свойство Time изменилось.

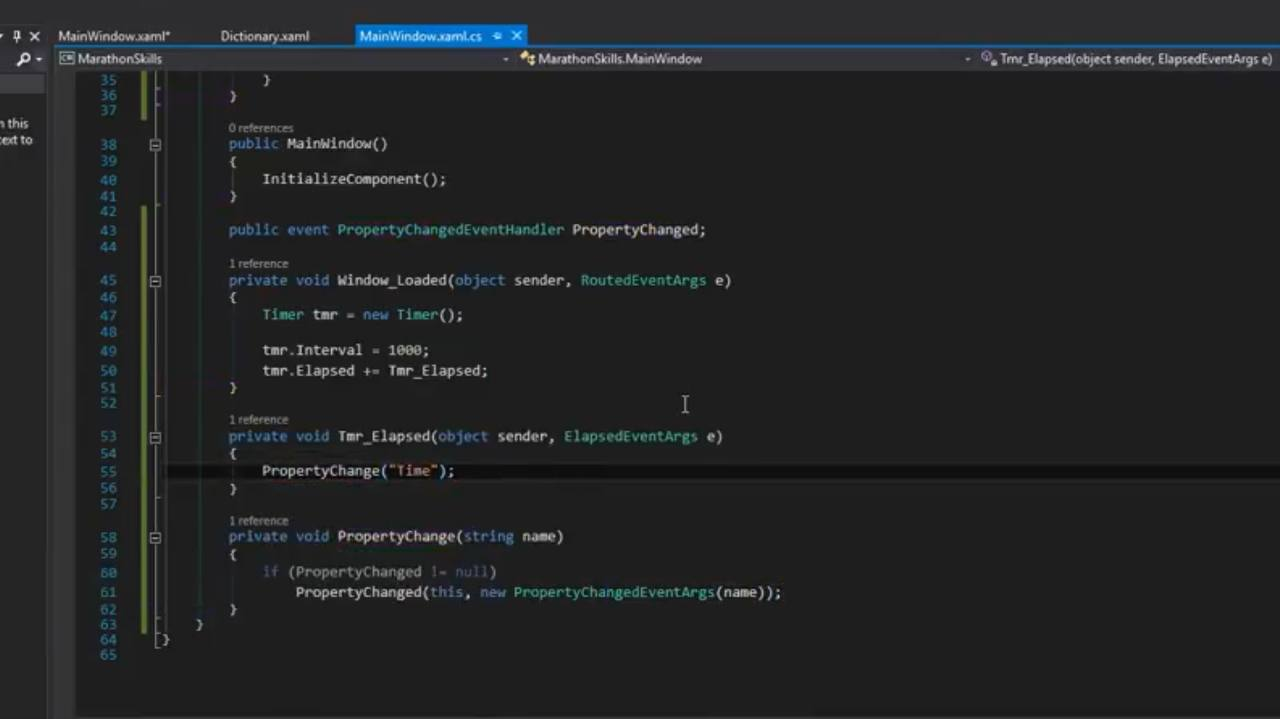


Рисунок 7. Функции Tmr\_Elapsed и PropertyChange.

В свойствах label, где прописан контент подписываем Binding к свойству времени, обновления которого происходят при изменим свойства. Полученные данные label не обновляет, а только получает.

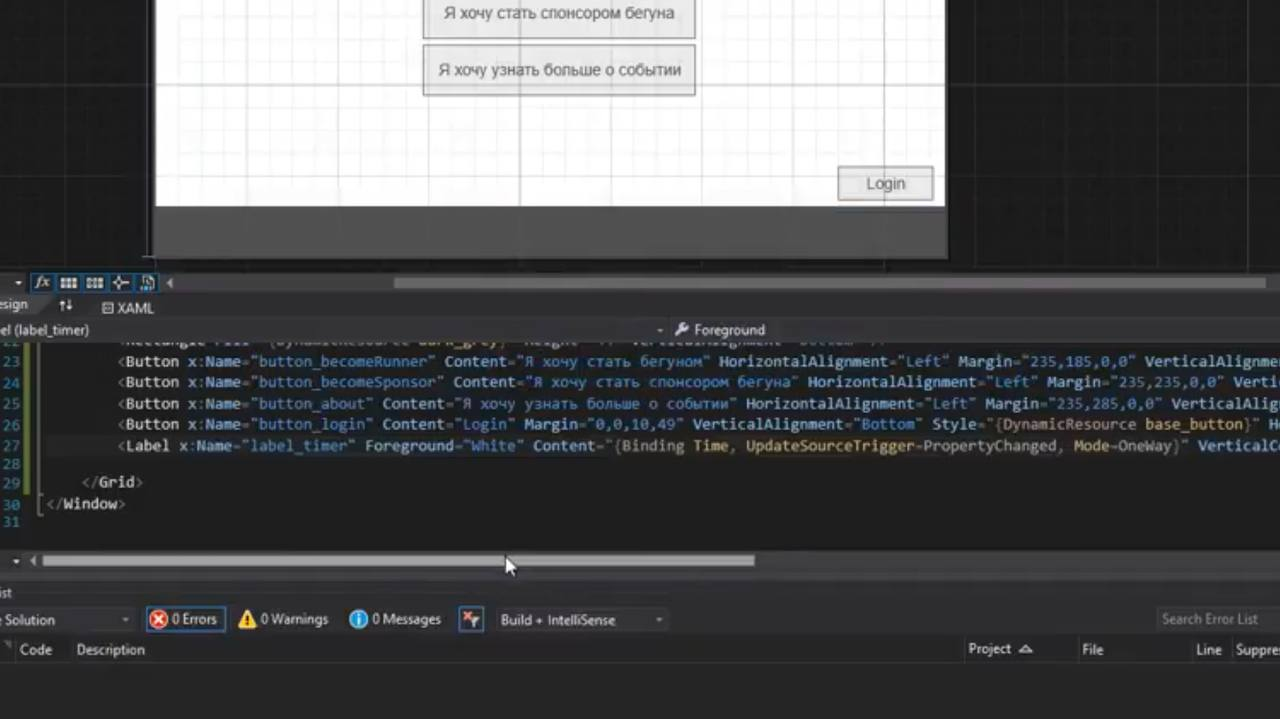


Рисунок 8. Свойство label для подключения таймера.

# **Подготовка для переходов на другие формы**

Для начала добавить новую страницу с названием MainPage. Укажем ее размеры: ширина 700, высота 420. Принести все главные элементы (кроме Rectangle и label) на новую страницу.

На главном окне добавить компонент frame, подогнать под главную форму. Указать в свойствах изначального источника MainPage, навигацию спрятать.

Из названия данной темы понятно, что данные действия были совершены для того, чтобы можно было осуществить переходы между формами. Но помимо этого, еще для того, чтобы можно было добавить кнопку назад.

# **Подключение базы данных**

*Для дальнейшей работы проект будет, перенес на виртуальную машину, поскольку в колледже недостаточно прав для подключения базы данных.*

Нажав пкм по проекту, добавим новый элемент в категории, данные с названием ADO.NET Entity Date Model. В названии указать model. В появившимся окне выбрать EF disigner from datadase. Это позволит нам создать модель из имеющийся базой данных. Выбираем создать подключение. Вписываем имя сервера и имя пользователя. Выбрать базу данных, созданную ранее. Проверяем соединение и ждем окей. Затем указываем название базы данных и ждем далее. В следующем пункте выбираем, что нам нужны только таблицы, и заканчиваем установку.

После того, как наша модель построится, должно появится следующее:

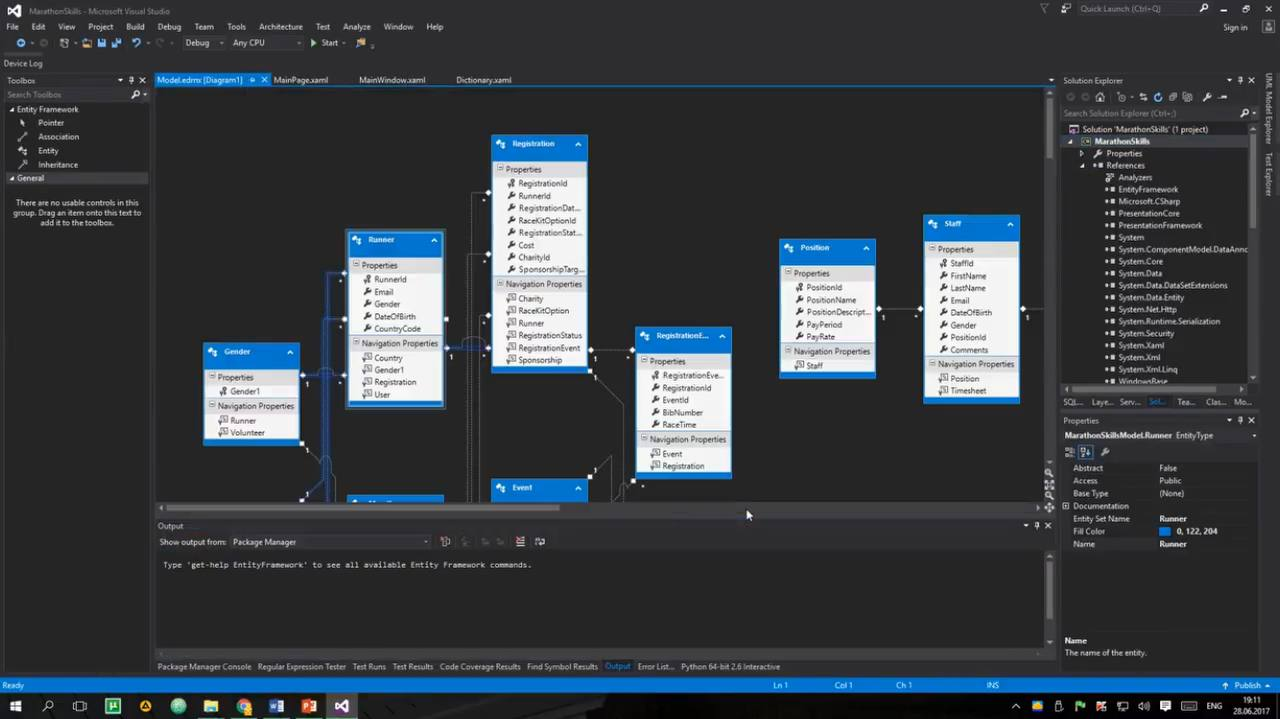


Рисунок 9. Созданная база данных.

# **Создание страницы спонсора бегуна**

# **Дизайн**

Создаем новую страницу с названием SponsorPage. Меняем размер окна: высота – 427, высота – 700.

Добавляем label. В свойствах в контенте пишем *"MARAFON SKILLS 2017"*, добавляем белый цвет текста и жирности. Задний фон делаем темно-серый через динамические ресурсы. Располагаем элементы по левой стороне *HorizontalContentAligment="left"*, и делаем отступ слева через свойство *Padding="100, 0, 0, 0".*

Добавляем кнопку с названием button\_back, в тексте прописываем *"Назад".*

Добавить RichTextBox. В свойствах указать прозрачный задний фон *Background="Transparent"*. Для RichTextBox создать стиль, в котором укажем шрифт Arial, обводку 0, возможность поменять текст, размер текста и цвет текста.

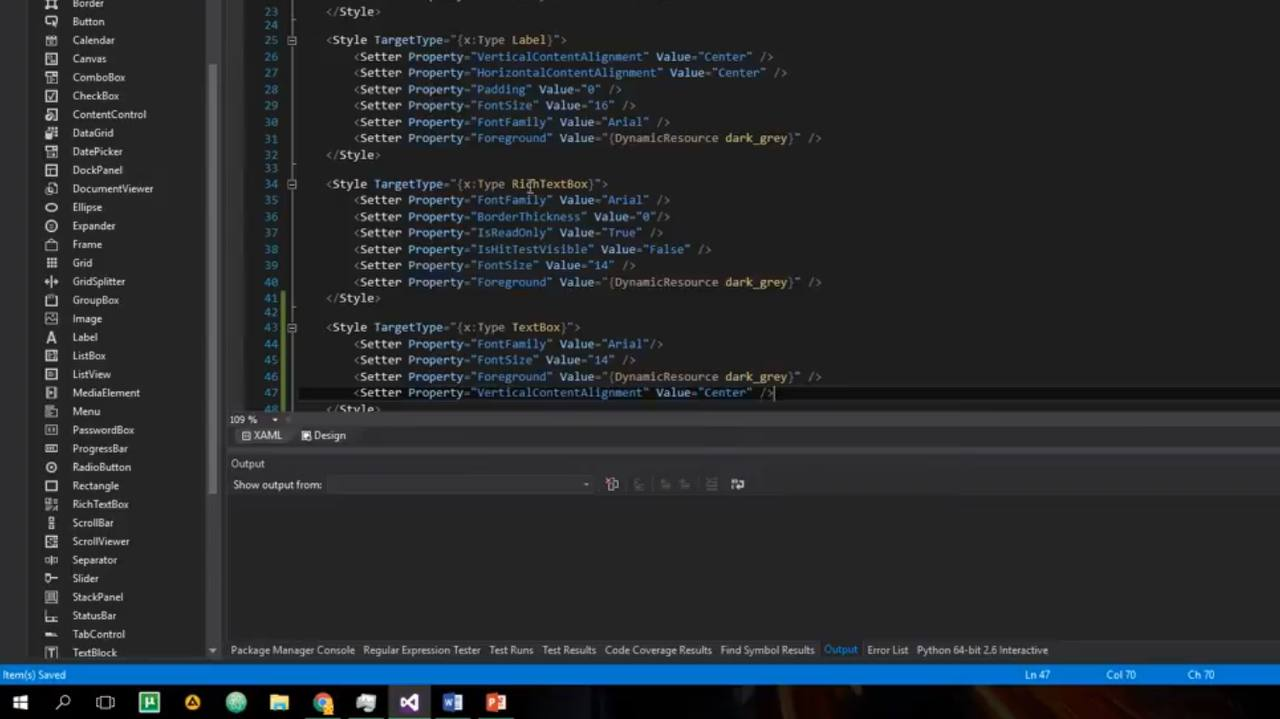


Рисунок 10. Стиль для RichTextBox.

В первом параграфе прописать *"Спонсор бегуна".*

Добавить второй параграф, в котором нужно прописать "*Пожалуйста, выберите бегуна, которого вы хотели бы спонсировать и сумму, которую вы хотели бы спонсировать. Спасибо за вашу поддержку бегунов и их благотворительных учреждений"*.

В свойствах документа добавить расположение текста по центру.

В свойствах первого параграфа добавить жирность текста.

Расположить RichTextBox в верхней части экрана.

Добавить новый label. Добавить текст *"Информация о спонсоре"*. Добавить жирность текста. Данный label расположить в левой части страницы.

Скопировать данный label и вставить. Новый label расположить, напротив, в правой части экрана. В контенте прописать *"Благотворительность"*.

Вставить еще один такой же label. Расположить его ниже благотворительности. В этом label написать *"Сумма пожертвования".*

Добавить две кнопки одинакового размера. На первой прописать текст *"Заплатить",* на второй *"Отмена".* Обе кнопки расположить снизу справа.

В словаре добавим стиль для любого TextBox. В стиле указываем шрифт, размер шрифта, цвет текста темно-серый, расположение контента по центру.

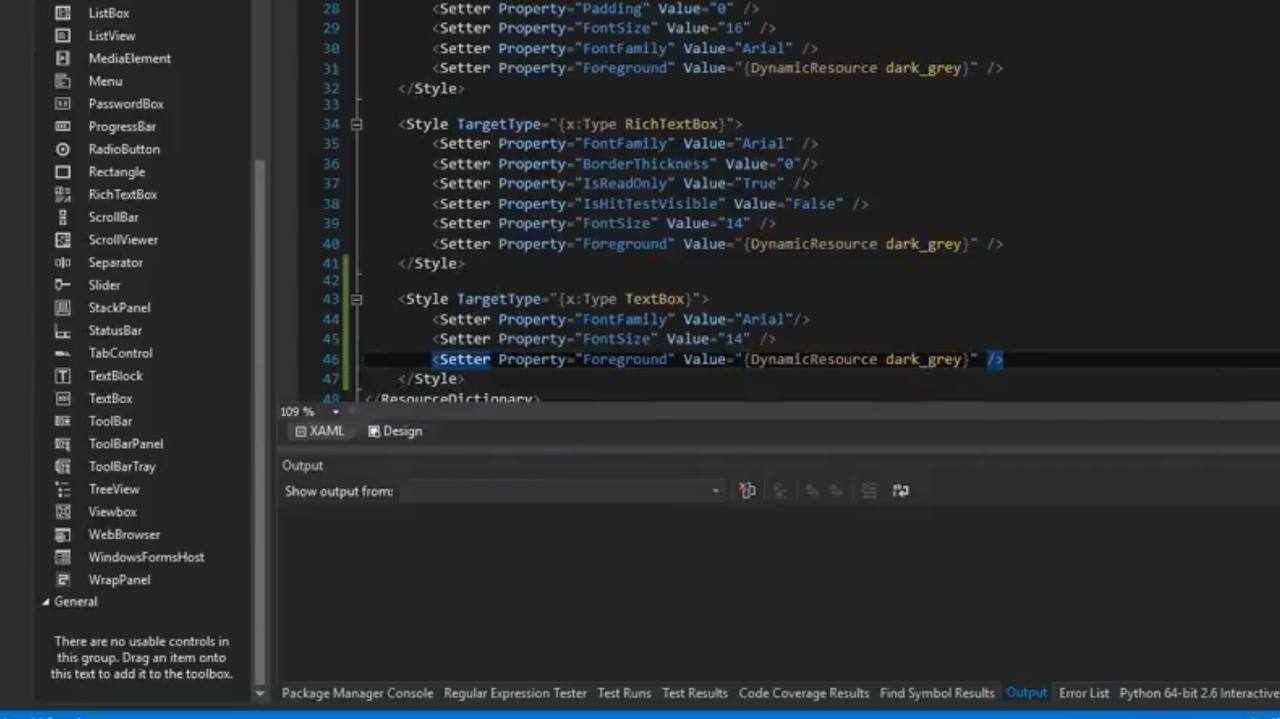


Рисунок 11. Стиль для TextBox.

Добавить TextBox. В имени указать textbox\_donation. В тексте указать "50" Также добавить свойство *HorizontalContentAligment="Center".* Его расположить под суммой пожертвования.

Добавить новый label. В названии указать label\_charity, в контенте прописать *"Фонд"*. Расположить под label с благотворительностью.

Добавить новый label. В названии указать label\_donation, в контенте *$ 50*. В свойствах добавить размер шрифта 30. Расположить под label с суммой пожертвования.

Слева от TextBox добавить кнопку, в контенте которого пишем "-", а в имени button\_sub. Копируем эту кнопку и вставляем справа от TextBox, в контенте которого пишем "+", а в имени button\_add.

Добавить новый label. В контенте указать *"Ваше имя:".* Напротив него создать TextBox, в имени которого указать textBox\_name. Данные элементы расположить ниже label с информацией о спонсоре.

Ниже добавить label, в контенте которого указать *"Бегун:".*

Напротив него создать ComboBox, в имени, которого указать comboBox\_runner.

Ниже добавить label, в контенте которого указать *"Карта:".* Напротив него создать TextBox, в имени которого указать textBox\_card.

Ниже добавить label, в контенте которого указать *"Номер карты:".* Напротив него создать TextBox, в имени которого указать textBox\_cardnum.

Ниже добавить label, в контенте которого указать *"Срок действия:".* Напротив него создать два TextBox, в имени первого указать textBox\_mon, в имени второго указать textBox\_year.

Ниже добавить label, в контенте которого указать *"CVC".* Напротив него создать TextBox, в имени, которого указать textBox\_cvc.

Для каждого из этих элементов в свойствах прописать *HorizontalContentAligment="right".*

# **Переход на другую форму**

Перейти на главную форму. Сделаем, обработчик событий для кнопки "Я хочу стать спонсором бегуна". Для этого в свойствах данной кнопки прописать следующее: *Click="button\_becomeSponsor\_Click".*

После этого переходим в Main.Page.xaml.cs и прописываем функцию *button\_becomeSponsor\_Click*. В ней прописываем данную строчку: *"NavigationService.Navigate(newUri("SponsorPage.xalm",UriKind.RelativeOrAdsolute)).*

Таким образом, мы организуем переход на другую форму.

Также сделаем рабочую кнопку назад и отмена, на странице спонсора.

Для этого в свойствах обеих кнопок прописываем *Click="button\_back\_Click".*

В SponsorPage.xalm.cs снова прописываем функцию button\_back\_Click, в которую прописываем следующую строчку: *NavigationService.GoBack();*

# **Загрузка бегунов в ComboBox**

Для этого нажмем на title и в панели свойств (в правом нижнем углу) найдем пункт Loaded. На нем кликаем два раза, затем нас перенесет в SponsorPage.xalm.cs.

После этого добавим новый класс подключения базы данных. С названием DB. Делаем этот класс публичным.

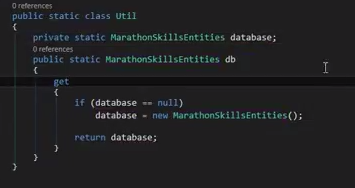


Рисунок 12. Связь класса с базой данных.

В MainWindow.xalm.cs нужно дописать следующую строчку:

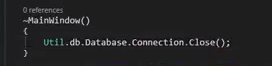


Рисунок 13.Связь с новым классом.

Затем добавим еще один класс, который будет содержать информацию только о бегунах. Его название RunnerInfo.

Теперь допишем функцию Page\_Loaded:

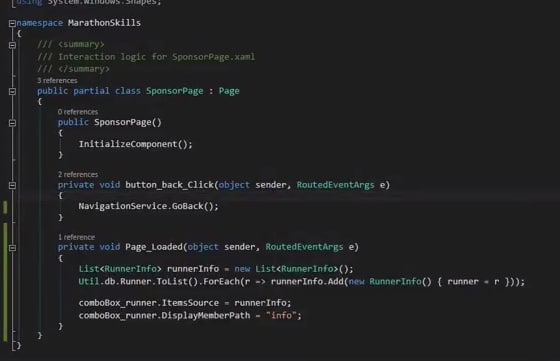


Рисунок 14. Код функции Page\_Loaded.

# **Реализация суммы пожертвования**

Для того чтобы реализовать сумму пожертвования, для кнопки с минусом прописываем свойство *Click="button\_sub\_click"*. Для кнопки с плюсом прописываем свойство *Click="button\_add\_click".*

После этого переходим в код (SponsorPage.xalm.cs). Прописываем публичное значение donation типа int и прописываем следующую функцию:

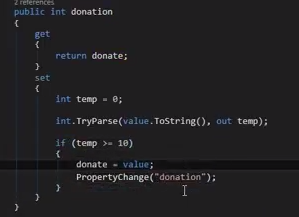


Рисунок 15. Код для переменной donation.

После этого в функции button\_sub\_Click прописываем цикл, в котором условие, если переменная donation больше или равна 20, то данная переменная уменьшается 10 и эти изменения записываются.

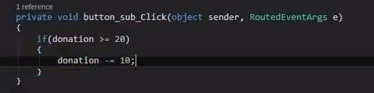


Рисунок 16. Функция для кнопки с минусом.

В свойствах label над суммой пожертвования, в контенте прописываем *"{Binding donation, UpdateSourceTrigger=LostFocus, Mode=OneWay".*

В свойствах TextBox, находящийся между кнопками минус и плюс, в тексте также прописываем *"{Binding donation, UpdateSourceTrigger=PropertyChanged, Mode=OneWay".*

Для функции button\_add\_Click, прописываем тоже самое, что и для другой кнопки, но с плюсом:



Рисунок 17. Функция для кнопки с плюсом.

# **Прогрузка фонда в зависимости от выбора бегуна**

В свойствах ComboBox прописать *SelectionChanged="comboBox\_runner\_SelectionChanged"*.

Переходим в код. В функции comboBox\_runner\_SelectionChanged прописываем следующее:

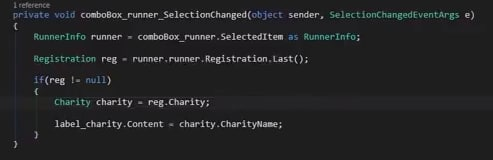


Рисунок 18. Функция для comboBox, связанная с базой данных.

Добавим новое окно с именем wndCharityInfo. В панели свойств ищем Resource и выбираем "(Collection)". Устанавливаем размер этого окна высота 376, ширина 478.

В title прописываем *"Благотворительная организация".*

Добавляем новый label, в имени которого указать "label\_charityName". Располагаем сверху.

Добавляем image, располагаем под label.

Добавляем textbox, располагаем снизу. В нем будет описание фонда. В свойствах прописываем тип шрифта Arial, через динамические ресурсы прописать цвет текста темно-серый.

Переходим в код этой страницы, прописываем следующее:

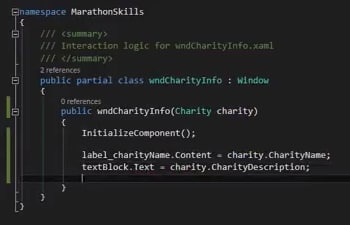


Рисунок 19. Функция для элементов страницы.

Этот код позволит нам автоматически брать информацию о фонде из базы данных, и добавлять ее на наше окно.

Переходим на страницу спонсора. В свойствах картинки пишем MouseDown="image\_MouseDown".

Переходим в код страницы спонсора. В функции image\_MouseDown прописываем следующее:

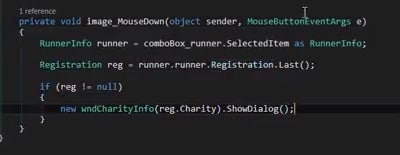


Рисунок 20. Функция для картинки.

Это позволит нам при нажатии на картинку открыть новое окно, в котором будет информация о фонде.

Функцию Page\_Loaded дополняем строчкой: *comboBox\_runner.SelectedIndex = 0*;

# **Проверка ввода информации и добавление ее в базу данных**

В этом пункте мы реализуем кнопку заплатить, а также осуществим проверку вводимой информации в textBox и добавление ее в базу данных.

Переходим на страницу спонсора. В свойствах кнопки прописываем *Click="button\_pay\_Click".*

Переходим в код этой страницы. В функции button\_pay\_Click прописываем следующий код:

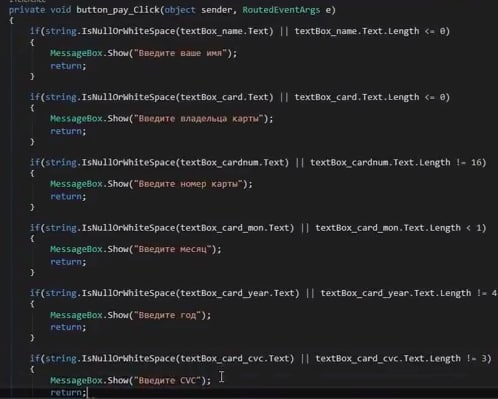


Рисунок 21. Функция для кнопки покупки.

Данный код будет выводить текст на textBox, который будет описать требуемую вводимую информацию.

Переходим снова на страницу спонсора. В свойствах textBox с картой прописываем *KeyDown="textBox\_cardnum\_KeyDown"*.

Снова переходим в код и в функции textBox\_cardnum\_KeyDown. В этой функции пропишем шаблон по заполнению этого textBox.

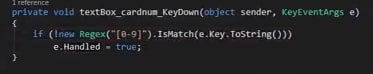


Рисунок 22. Функция для textBox.

На странице спонсора в свойствах textBox с месяцем, годом, CVC и суммой пожертвования соответственно прописываем следующую команду: *"textBox\_cardnum\_KeyDown".*

Теперь дополним нашу функцию button\_pay\_Click:

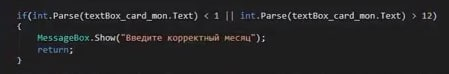


Рисунок 23. Дополнена функция button\_pay\_Click.

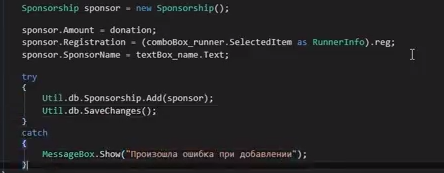


Рисунок 24. Функция для проверки введенных данных.

Данный код будет проверять на правильность введенных данных, и выдать ошибку.

Дополним функцию Page\_Loaded для болей точности работы кода:

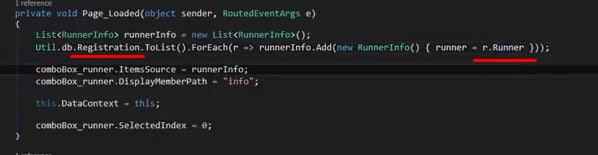


Рисунок 25. Дописана функция Page\_Loaded.

# **Форма для подтверждения оплаты**

# **Дизайн**

Создаем новую страницу с именем SponsorshipSuccess. Указываем его размеры: высота 427, ширина 700.

С предыдущей страницы копируем верхнею панель и добавляем на эту страницу.

Добавляем RichTextBox. В первом параграфе пишем *"Спасибо за вашу спонсорскую поддержку!"*.

Во втором параграфе *"Спасибо за поддержку бегуна в Marathon Skills 2016! Ваше пожертвование пойдет в его благотворительную организацию".* В свойствах документа располагаем текст по центру,

Добавляем новый label. Добавляем ему имя label\_runner.

Ниже располагаем новый label. В имени указываем label\_charity.

Ниже добавляем еще один label. В имени указываем label\_amout. Размер шрифта ставим 30.

Добавляем кнопку посередине экрана. В его имени указываем button\_success. Также в свойствах прописываем *Click="button\_back\_Click".*

Переходим в код данной страницы. В функции button\_back\_Click прописываем код, который при нажатии на кнопку вернет нас на главную страницу:

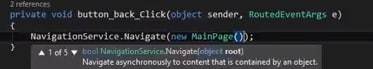


Рисунок 26. Функция для возвращения на предыдущею страницу.

В коде данной страницы прописываем, информацию про label. А именно, чтобы в них указывалась информация, указанная на предыдущей странице.

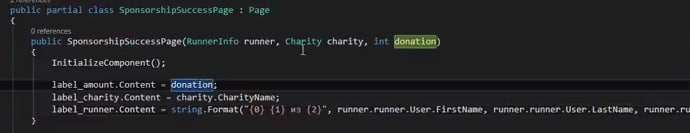


Рисунок 27. Функция для информации.

Теперь дополним нашу функцию button\_pay\_Click в коде страницы спонсора. Здесь мы пропишем код, который перекинет нас на другую страницу в случае успешной оплаты.

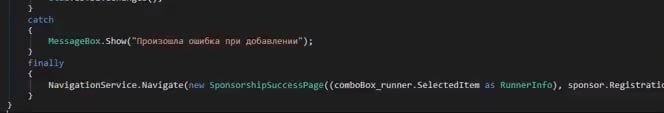


Рисунок 28. Законченный код button\_pay\_Click.

# **Заключение**

Передо мной стояла задача создания рабочего приложения Marafon Skills 2017, и как я считаю я выполнила данное задание. Приложение содержит в себе главное меню, спонсирование проекта, собственную базу данных. Также мне удалось сделать подробную инструкцию по созданию данного приложения, которое может создать любой человек.

Рабочее приложение я добавила на GitHub, где любой желающий может попробовать его самостоятельно.

# **Программные коды проекта.**

**Программный код для app.xalm.**

<Application x:Class="marafon\_skills.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:marafon\_skills"

StartupUri="MainWindow.xaml">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/Resource/Dictionary.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</Application>

**Программный код для класса DB.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace marafon\_skills

{

public static class Util

{

private static MarathonSkillsEntities database;

public static MarathonSkillsEntities db

{

get

{

if (database == null)

database = new MarathonSkillsEntities();

return database;

}

}

}

**Программный код для класса MainPage.xalm. cs.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace marafon\_skills

{

public partial class MainPage : Page

{

public MainPage()

{

InitializeComponent();

}

private void richTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button\_becomeSponsor\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Uri("SponsorPage.xaml", UriKind.RelativeOrAbsolute));

}

private void Button\_about\_Click(object sender, RoutedEventArgs e)

{

}

}

}

**Программный код для класса MainWindow.xalm.cs.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Timers;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Timer = System.Timers.Timer;

namespace marafon\_skills

{

public partial class MainWindow : Window, INotifyPropertyChanged

{

public string Time

{

get

{

DateTime dt1 = DateTime.Now;

DateTime dt2 = DateTime.Parse("2023-10-21 6:00");

TimeSpan ts = dt2 - dt1;

return string.Format("{0} дн {1} ч {2} мин {3} сек до старта марафона!", ts.Days, ts.Hours, ts.Minutes, ts.Seconds);

}

}

public MainWindow()

{

InitializeComponent();

this.DataContext = this;

}

~MainWindow()

{

Util.db.Database.Connection.Close();

}

public event PropertyChangedEventHandler PropertyChanged;

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

Timer tmr = new Timer();

tmr.Interval = 1000;

tmr.Elapsed += Tmr\_Elapsed;

tmr.Start();

}

private void Tmr\_Elapsed(object sender, ElapsedEventArgs e)

{

PropertyChange("Time");

}

private void PropertyChange(string name)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(name));

}

private void RichTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

throw new NotImplementedException();

}

private void frame\_Navigated(object sender, NavigationEventArgs e)

{

}

}

}

**Программный код для словаря Resource.**

<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:marafon\_skills">

<SolidColorBrush x:Key="light\_grey" Color=" #EBEBEB"/>

<SolidColorBrush x:Key="grey" Color=" #B4B4B4"/>

<SolidColorBrush x:Key="dark\_grey" Color=" #505050"/>

<Style TargetType="{x:Type Button}">

<Setter Property="Background" Value="{DynamicResource light\_grey }" />

<Setter Property="FontSize" Value="14" />

<Setter Property="FontFamily" Value="Arial" />

<Setter Property="Height" Value="30" />

<Setter Property="Foreground" Value="{DynamicResource dark\_grey }" />

</Style>

<Style x:Key="base\_button" TargetType="Button" >

<Setter Property="Background" Value="{DynamicResource light\_grey }" />

<Setter Property="FontSize" Value="14" />

<Setter Property="FontFamily" Value="Arial" />

<Setter Property="Height" Value="10" />

<Setter Property="Foreground" Value="{DynamicResource dark\_grey }" />

</Style>

<Style x:Key="base\_window" TargetType="Window">

<Setter Property="Width" Value="700" />

<Setter Property="Height" Value="500" />

<Setter Property="ResizeMode" Value="CanMinimize" />

<Setter Property="FontSize" Value="14" />

<Setter Property="FontFamily" Value="Arial" />

</Style>

<Style TargetType="{x:Type Label}">

<Setter Property="VerticalContentAlignment" Value="Center" />

<Setter Property="HorizontalContentAlignment" Value="Center" />

<Setter Property="Padding" Value="0" />

<Setter Property="FontSize" Value="16" />

<Setter Property="FontFamily" Value="Arial" />

<Setter Property="Foreground" Value="{DynamicResource dark\_grey }" />

</Style>

<Style TargetType="{x:Type RichTextBox}">

<Setter Property="FontFamily" Value="Arial" />

<Setter Property="BorderThickness" Value="0" />

<Setter Property="IsReadOnly" Value="True" />

<Setter Property="IsHitTestVisible" Value="False" />

<Setter Property="FontSize" Value="14" />

<Setter Property="Foreground" Value="{DynamicResource dark\_grey }" />

</Style>

<Style TargetType="{x:Type TextBox}">

<Setter Property="FontFamily" Value="Arial" />

<Setter Property="FontSize" Value="14" />

<Setter Property="Foreground" Value="{DynamicResource dark\_grey }" />

<Setter Property="VerticalContentAlignment" Value="Center" />

</Style>

</ResourceDictionary>

**Программный код для класса RunnerInfo.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace marafon\_skills

{

internal class RunnerInfo

{

public Runner runner { get; set; }

public string info

{

get

{

return string.Format(" {0} {1} - {2} {3}", runner.User.FirstName, runner.User.LastName, runner.CountryCode, runner.Country.CountryName);

}

}

}

}

**Программный код для SponsorPage.xalm.cs.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace marafon\_skills

{

/// <summary>

/// Логика взаимодействия для SponsorPage.xaml

/// </summary>

public partial class SponsorPage : Page, INotifyPropertyChanged

{

private int donate = 50;

public int donation

{

get

{

return donate;

}

set

{

int temp = 0;

int.TryParse(value.ToString(), out temp);

donate = value;

if (temp >= 10)

{

donate = value;

PropertyChange("donation");

}

}

}

public SponsorPage()

{

InitializeComponent();

}

public event PropertyChangedEventHandler PropertyChanged;

private void PropertyChange(string name)

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(name));

}

private void button\_cancel\_Click(object sender, RoutedEventArgs e)

{

}

private void button\_back\_Click(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

private void Page\_Loaded(object sender, RoutedEventArgs e)

{

List<RunnerInfo> runnerInfo = new List<RunnerInfo>();

Util.db.Runner.ToList().ForEach(r => runnerInfo.Add(new RunnerInfo() { runner = r }));

comboBox\_runner.ItemsSource = runnerInfo;

comboBox\_runner.DisplayMemberPath = "info";

this.DataContext = this;

}

private void richTextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void TextBox\_donation\_TextChanged(object sender, TextChangedEventArgs e)

{

}

private void button\_add\_Click(object sender, RoutedEventArgs e)

{

donation += 10;

PropertyChange("donation");

}

private void button\_sub\_Click(object sender, RoutedEventArgs e)

{

if(donation >= 20)

donation -= 10;

PropertyChange("donation");

}

}

}

# **Список источников**

*Все видео, используемые для этого проекта и сам проект залит на GitHub, как и собственно этот отчет.*