Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина: Программирование (П)

Пояснительная записка к курсовой работе

Тема работы:

«Приложение для поиска ресторанов»

Исполнитель

студентка гр. 653501 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Гуринович Д.А.

(подпись дата)

Руководитель

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Козуб В.Н.

(подпись дата)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка)

Минск 2017

Содержание

[Введение](#_Toc480311999) 3

[1. Сравнение с аналогичными продуктами](#_Toc480312000) 4

[2. Windows Store-приложения](#_Toc480312001) 5

[3. Пользовательский интерфейс](#_Toc480312002) 6

[4. Технические аспекты разработки приложения](#_Toc480312003) 13

[4.1. Технологии и их использование в работе приложения](#_Toc480312004) 13

[4.2. Описание работы приложения](#_Toc480312005) 23

[Заключение](#_Toc480312024) 24

Приложение 1. Язык разметки XAML25

[Список использованных источников](#_Toc480312025) 30

**Введение**

Посещение кафе и ресторанов является неотъемлемой частью нашей жизни. Бывают моменты, когда необходимо либо срочно найти место рядом, либо просто выбрать заведение, подходящее по ценам и по составляющей меню.

Данное приложение (Café Guide) поможет пользователю решить эту проблему. Оно определяет по местоположению ближайшие кафе и рестораны. С помощью приложения пользователь также узнает меню заведения, его рейтинг и время работы. Благодаря этим данным определить, какое кафе или ресторан стоит посетить, становится намного проще.

Таким образом, целью данной курсовой работы является создание программного средства для поиска заведений в радиусе километра от выбранной на карте точки, которое поможет пользователю получить основные сведения о найденных кафе и ресторанах.

Для достижения цели необходимо решить следующие задачи:

1. Разработать программное средство на платформе WinRT на языке C#, способное определять местоположение пользователя, находить ближайшие заведения и информацию о них;
2. Реализовать понятный и удобный для пользователя интерфейс программного средства с помощью языка разметки XAML.

**1. Сравнение с аналогичными продуктами**

Сейчас для пользователей в открытом доступе существует множество карт с огромной базой данных. Возьмем для сравнения один из самых крупнейших сервисов Google Maps.

Google Maps - набор приложений, построенных на основе бесплатного картографического сервиса и технологии, предоставляемых компанией Google. Многие пользователи нехотя пользуются Google Maps, чтобы найти ближайшие заведения, т.к. это не является основным функционалом приложения, из-за чего им становится очень сложно разобраться, как получить нужную информацию. Также, например, при запросе в Google Maps «Рестораны» на карте отображаются не ближайшие места, а места с лучшим рейтингом. Т.е. ближайшие заведения не будут отображены на карте. Таким образом, используя Google Maps, очень неудобно находить ближайшие кафе и рестораны.

Самым популярным приложением в этой области является Foursquare. Foursquare - приложение для локального поиска и обнаружения, предоставляющее результаты своим пользователям. Однако оно, как и Google Maps, не ориентировано только на поиск ресторанов или кафе, его функционал намного больше, поэтому опять же пользователям неудобно искать такие заведения. Также в Foursquare нет подробной информации о месте, что не дает пользователю возможности получить необходимые ему сведения (например, работает ли сейчас кафе).

Таким образом, в отличие от Google Maps и Foursquare, Café Guide ориентирован на поиск только кафе и ресторанов, что облегчает взаимодействие пользователя с приложением. Также Café Guide содержит такую информацию о заведении, как его меню, рейтинг и время работы, что поможет пользователю определиться с выбором.

**2.** **Windows Store-приложения**

По типу программной платформы приложения для Windows можно разделить на две группы:

1. Классические Windows-приложения.

2. Windows Store-приложения.

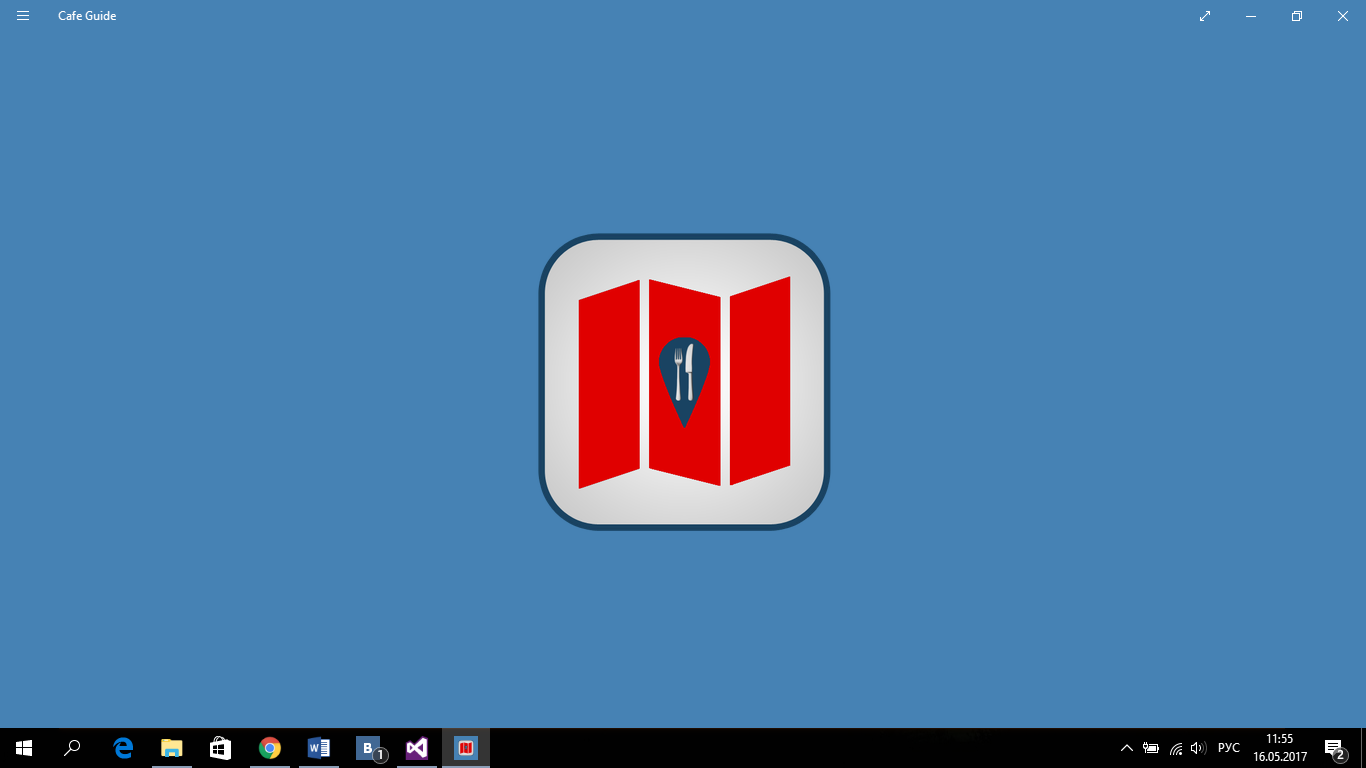
Windows Runtime — это новая модель разработки приложений, появившаяся, начиная с Windows 8, а также объектно-ориентированный языконезависимый программный интерфейс (API), написанный на неуправляемом коде и реализующий концепции асинхронного программирования. Все функции и методы, потенциально работающие более 50 мс, реализованы асинхронно. Синхронных аналогов для них нет. Это обеспечивает лучшие характеристики и бóльшую "отзывчивость" приложений.

Windows Store-приложения – приложения, использующие программную платформу WinRT. Windows Store-приложения могут создаваться на различных языках программирования: C#, JavaScript, Visual Basic и C++. Пользовательский интерфейс приложений, написанных на JavaScript, создается с помощью HTML 5, в остальных случаях применяется XAML (eXtensible Application Markup Language — расширяемый язык разметки приложений). XAML стал и частью Windows Runtime, а значит, и одним из базовых компонентов операционной системы Windows. Архитектура платформы Windows 8:



**3. Пользовательский интерфейс**

Как и любое приложение, Café Guide начинается с логотипа:



Логотип представляет собой карту, на которой имеется указатель на кафе или ресторан (вилка и нож). Логотип приложения является и его иконкой, и заставкой.

Пользовательский интерфейс реализован с помощью языка разметки XAML. (подробнее о XAML в приложении 1).

Приложение Café Guide состоит из двух страниц: основной и информационной.

Элементы основной страницы размещены с помощью менеджера Grid, состоящего из одной строки и двух столбцов. В первом столбце располагается карта, а во втором – элементы управления приложения. В листинге 3.1 показано размещение элементов в Grid.

***Листинг 3.1 Размещение элементов в Grid***

<Grid Name="MainGrid" HorizontalAlignment="Stretch">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="Auto" />

</Grid.ColumnDefinitions>

<bing:Map ELEMENT\_PROPERTIES\_HERE/>

<StackPanel ELEMENT\_PROPERTIES\_HERE>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBox ELEMENT\_PROPERTIES\_HERE/>

<Button ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBox ELEMENT\_PROPERTIES\_HERE/>

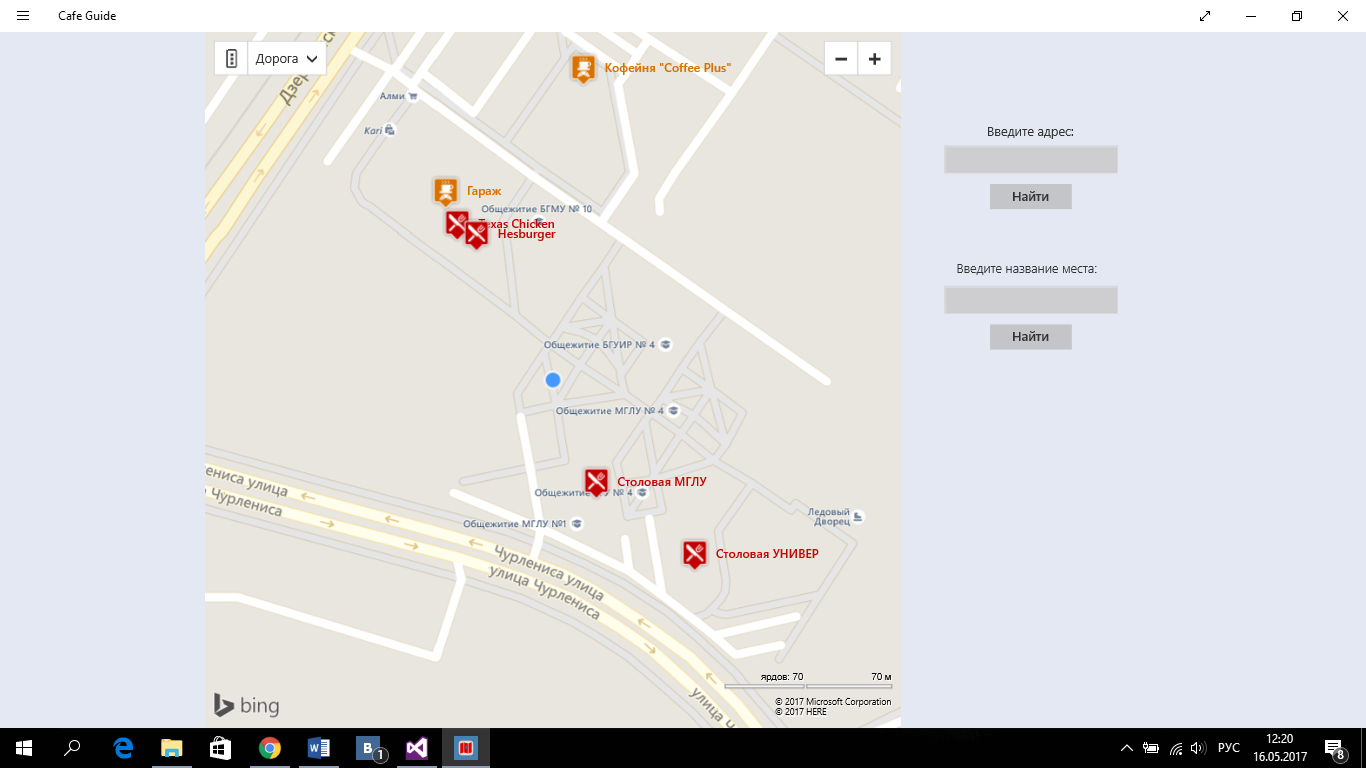
<Button ELEMENT\_PROPERTIES\_HERE/>

<Button ELEMENT\_PROPERTIES\_HERE/>

</StackPanel>

</Grid>

Интерфейс основной страницы выглядит таким образом:



Как можно заметить, страница содержит карту с маркерами, поле для поиска адреса и поле для поиска заведения по его названию.

Элемент управления для отображения карт не входит в поставку Windows 8. Для реализации приложения используется Bing Maps SDK for Windows Store apps. Это дает возможность использовать элемент управления Map. При запуске приложения Map обращается к сервису карт Bing, и для его использования необходим ключ, который указывается в свойстве Credentials элемента управления Map. В листинге 3.2 показан элемент управления Map.

***Листинг 3.2 Элемент управления Map***

<bing:Map Name="MyMap" Margin="0,0,0,0" Height="800" Width="800"

ZoomLevel="7" Credentials="YOUR\_BING\_MAP\_KEY"/>

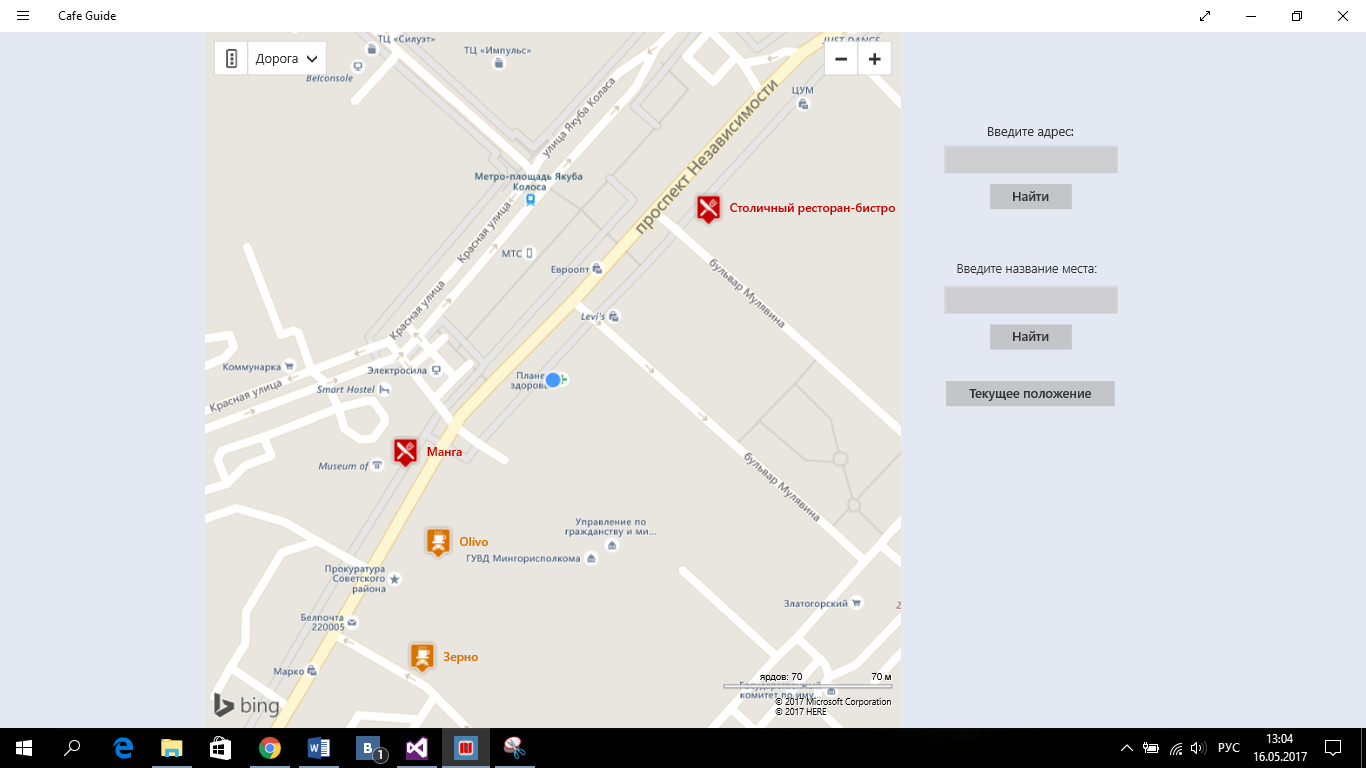
На карте размещены маркеры трех типов:

 , обозначающий текущее положение пользователя;

 , обозначающий кафе;

, обозначающий ресторан.

Если выполнить поиск адреса, то маркер будет показывать адрес, введенный пользователем, а маркеры и ближайшие к нему кафе и рестораны. Пример поискового запроса по адресу пр. Независимости, 48:



Также можно заметить, что после выполнения поискового запроса появляется кнопка «Текущее положение», которая возвращает пользователя к текущему положению. Кнопка «Текущее положение» показана в листинге 3.3.

***Листинг 3.3 Элемент управления кнопка «Текущее положение»***

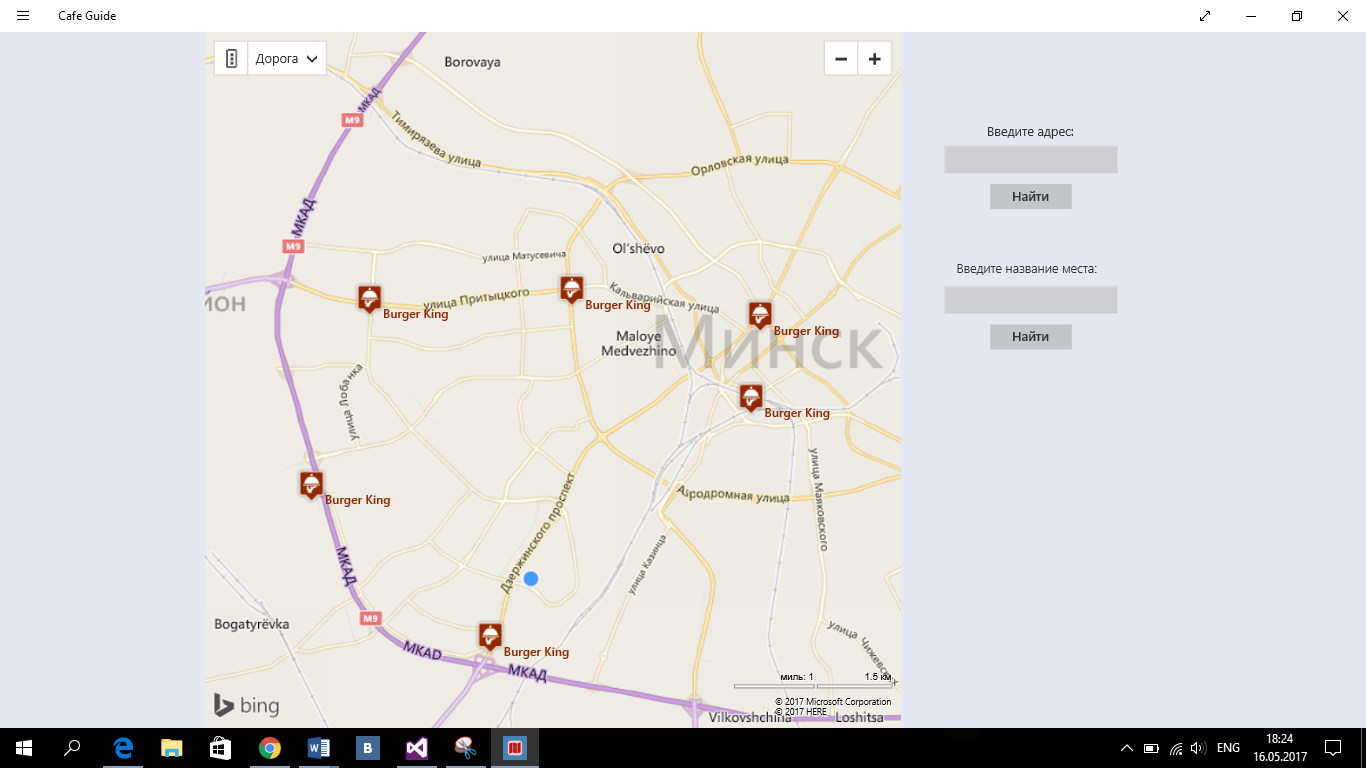
<Button Name="ReturnToTheCurrentLocation" Content="Текущее положение"

Width="200" BorderBrush="#FF323232" Foreground="#FF323232"

HorizontalAlignment="Center" Margin="0,30,0,0"

Visibility="Collapsed" Click="ClickedReturnToTheCurrentLocation"/>

Если вы полнить поиск заведения, то оно будет отмечено маркером на карте. Пример поискового запроса по названию Burger King:



При нажатии на название любого кафе или ресторана приложение переходит на информационную страницу:



Элементы информационной страницы также размещены с помощью нескольких менеджеров Grid и StackPanel. В листинге 3.4 показано размещение элементов на информационной странице.

***Листинг 3.4 Размещение элементов на информационной странице***

<Grid Background="#FFE3E8F3">

<Grid.ChildrenTransitions>

<TransitionCollection>

<EntranceThemeTransition/>

</TransitionCollection>

</Grid.ChildrenTransitions>

<Grid.RowDefinitions>

<RowDefinition Height="140"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="120"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Button ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

</Grid>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width = "\*" MinWidth="400"/>

<ColumnDefinition Width = "\*" MinWidth="200"/>

<ColumnDefinition Width = "\*" MinWidth="400"/>

</Grid.ColumnDefinitions>

<Grid Grid.Column="0" Margin="20,0,0,0">

<Grid.RowDefinitions>

<RowDefinition Height="400"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

<RowDefinition Height="auto"/>

</Grid.RowDefinitions>

<Image ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<HyperlinkButton ELEMENT\_PROPERTIES\_HERE/>

</Grid>

<StackPanel Grid.Column="1" Margin="0,50,0,0" >

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

</StackPanel>

<Grid Grid.Column="2" Margin="20,0,0,0" Name="MenuGrid">

<Grid.RowDefinitions>

<RowDefinition Height = "auto"/>

<RowDefinition Height = "\*"/>

</Grid.RowDefinitions>

<TextBlock ELEMENT\_PROPERTIES\_HERE/>

<Grid Grid.Row="1">

<Grid.ColumnDefinitions>

<ColumnDefinition Width = "auto"/>

<ColumnDefinition Width = "auto" MinWidth="100"/>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Name="DishStackPanel"

MinWidth="300" Margin="0,20,0,0"/>

<StackPanel Grid.Column="1" Name="PriceStackPanel"

MinWidth="100" Margin="0,20,0,0"/>

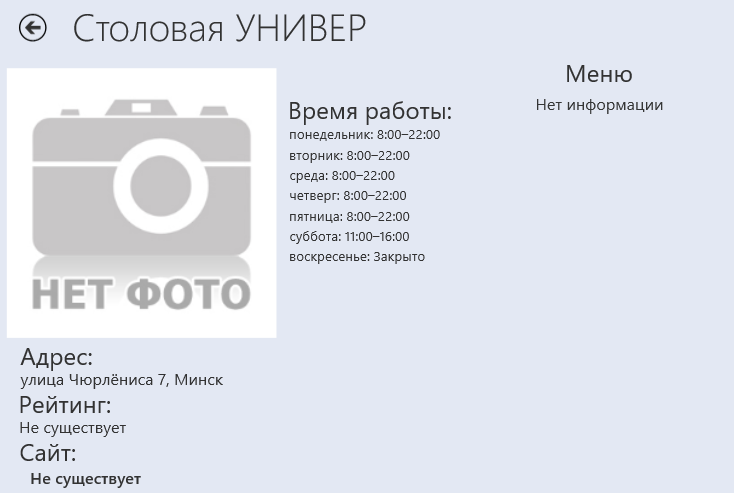
</Grid>

</Grid>

</Grid>

</Grid>

Как можно заметить, информационная страница содержит название кафе или ресторана, его адрес, рейтинг, сайт, фото, время работы и меню. Меню включает в себя название блюда и его цену. При нажатии на URL адрес сайта, он будет открыт в веб-браузере пользователя. Если какая-то информация о месте отсутствует, то это будет отображено на странице:



**4.** **Технические аспекты разработки приложения**

**4.1. Технологии и их использование в работе приложения**

Для реализации приложения необходимо было решить такие проблемы, как получение информации о местоположении пользователя, определение координат места по его адресу или названию и получение необходимых сведений о заведении.

Для определения местоположения пользователя используется пространство имен C# Windows.Devices.Geolocation. В приложении Café Guide местоположение определяется с помощью асинхронного метода IdentifyMyCurrentLocation():

private async void IdentifyMyCurrentLocation()

{

Geolocator geolocator = new Geolocator();

Geoposition geoposition = await geolocator.GetGeopositionAsync();

\_latitude = geoposition.Coordinate.Point.Position.Latitude;

\_longitude = geoposition.Coordinate.Point.Position.Longitude;

if (\_geolocator != null) \_geolocator.PositionChanged -= GeolocatorPositionChanged;

\_geolocator = new Geolocator();

\_geolocator.MovementThreshold = 100;

\_geolocator.ReportInterval = 60 \* 100;

\_geolocator.PositionChanged += GeolocatorPositionChanged;

//Очищает слой карты с отметками заведений

\_dataLayer.Children.Clear();

FindTheNearestPlaces();

}

Метод находит не только текущее местоположение пользователя, но и следит за его передвижением. Чтобы проверять текущую координату через определенный промежуток времени с помощью таймера, класс Geolocator предоставляет удобный способ —подписаться на событие PositionChange. Расстояние, при перемещении на которое относительно предыдущего вызова будет генерироваться следующий вызов, может быть контролируемо. Для этого необходимо установить значение свойства MovementThreshold (в метрах) объекта класса Geolocator. Сервис определения местоположения через определенные промежутки времени вычисляет координаты независимо от значения свойства MovementThreshold. После этого проверяется изменение координат относительно предыдущего значения и, если необходимо, генерируется событие PositionChanged. Для оптимизации энергопотребления необходимо задать свойство ReportInterval, которое определяет интервал времени в миллисекундах, через который будут вычисляться новые координаты.

Google обладает большой базой данных, поэтому было решено пользоваться именно ей для решения остальных проблем.

Геокодирование - это процесс преобразования адреса в географические координаты. Для выполнения геокодирования был использован Google Maps Geocoding API. Чтобы получить координаты места по его адресу, необходимо выполнить запрос:  
https://maps.googleapis.com/maps/api/geocode/xml?address=Minsk&key=***YOUR\_API\_KEY***, где xml – формат, в котором получать ответ, address – адрес места, key – ключ Google API. В приложении Café Guide это выполняет событие ClickedSearchAddress(object sender, RoutedEventArgs e). Оно не только делает запрос в Google, но и получает координаты места и делает отметку на карте. Код события:

private async void ClickedSearchAddress(object sender, RoutedEventArgs e)

{

//Выводится ошибка о том, что адрес не был введен

if (AddressName.Text == "")

{

MessageDialog msgDialog = new MessageDialog("Сначала введите адрес", "Ошибка");

UICommand okBtn = new UICommand("OK");

msgDialog.Commands.Add(okBtn);

await msgDialog.ShowAsync();

}

else

{

var address = AddressName.Text;

AddressName.Text = "";

var requestUri = string.Format("http://maps.googleapis.com/maps/api/geocode/xml?address={0}&sensor=false", Uri.EscapeDataString(address));

//Отправка запроса и получение ответа в формате XML

var request = WebRequest.Create(requestUri);

var response = await request.GetResponseAsync();

var xdoc = XDocument.Load(response.GetResponseStream());

//Анализ полученного ответа

var status = xdoc.Element("GeocodeResponse").Element("status");

if (status.Value == "OK")

{

//Получение широты и долготы из ответа

var result = xdoc.Element("GeocodeResponse").Element("result");

var locationElement = result.Element("geometry").Element("location");

var lat = locationElement.Element("lat");

var lng = locationElement.Element("lng");

\_latitude = double.Parse(lat.Value.Replace('.', ','));

\_longitude = double.Parse(lng.Value.Replace('.', ','));

\_dataLayer.Children.Clear();

//Добавление метки на карту

AddPushpin(\_latitude, \_longitude);

//Поиск ближайших мест

FindTheNearestPlaces();

}

else

{ //Ошибка, если ответ не был получен

MessageDialog msgDialog = new MessageDialog("Невозможно найти адрес", "Ошибка!");

UICommand okBtn = new UICommand("OK");

msgDialog.Commands.Add(okBtn);

await msgDialog.ShowAsync();

}

//Сделать кнопку "Текущее положение" видимой

ReturnToTheCurrentLocation.Visibility = Visibility.Visible;

}

}

Для получения координат ближайших кафе и ресторанов и информации о них используется The Google Places API Web Service. Чтобы получить координаты ближайших заведений, необходимо выполнить запрос:

https://maps.googleapis.com/maps/api/place/nearbysearch/output?parameters, где output?parameters должны содержать формат возвращаемого ответа, ключ Google API, координаты места, около которого осуществляется поиск, (широта и долгота) и радиус поиска. Чтобы найти место по его названию, достаточно просто добавить в output?parameters ключевое слово, что и является названием заведения.

В приложении Café Guide это выполняет асинхронный метод FindThePlaces(string stringRequest, string pictureSource, Color color):

private async void FindThePlaces(string stringRequest, string pictureSource, Color color)

{

//Загружает метку, которой отмечать место

var myImage = new BitmapImage(new Uri(pictureSource));

//Отправка запроса и получение ответа в формате XML

string baseUri = stringRequest;

var request = WebRequest.Create(baseUri);

var response = await request.GetResponseAsync();

var xDoc = XDocument.Load(response.GetResponseStream());

//Анализ ответа

var status = xDoc.Element("PlaceSearchResponse").Element("status");

if (status.Value == "OK")

{

IEnumerable<XElement> resultElement = from element in

xDoc.Element("PlaceSearchResponse").Elements("result")

select element;

//Для каждого результата отметить его на карте

foreach (XElement element in resultElement)

{

string placeName = element.Element("name").Value;

string placeId = element.Element("place\_id").Value;

double placeLatitude = double.Parse(element.Element("geometry").Element("location").

Element("lat").Value.Replace('.', ','));

double placeLongitude = double.Parse(element.Element("geometry").Element("location").

Element("lng").Value.Replace('.', ','));

//Функция добавления места на карту

AddPlace(placeName, placeId, myImage, placeLatitude, placeLongitude, color);

}

}

else

{

//Ошибка, если не было ничего найдено

MessageDialog msgDialog = new MessageDialog("К сожалению, мы не можем ничего найти", "Ошибка");

UICommand okBtn = new UICommand("OK");

msgDialog.Commands.Add(okBtn);

await msgDialog.ShowAsync();

}

}

В функции AddPlace(placeName, placeId, myImage, placeLatitude, placeLongitude, color) на карту добавляется элемент HyperlinkButton, при взаимодействии с которым приложение переходит на информационную страницу. Исходный код метода:

private void AddPlace(string placeName, string placeId, BitmapImage myImage, double placeLatitude, double placeLongitude, Color color)

{

//Создание подписи названия заведения

var placeInfo = new HyperlinkButton();

placeInfo.Content = placeName;

placeInfo.Foreground = new SolidColorBrush(color);

placeInfo.FontWeight = FontWeights.SemiBold;

placeInfo.FontSize = 14;

//Название HyperlinkButton ID места

placeInfo.Name = placeId;

//Создание события HyperlinkButton

placeInfo.Click += ClickedHyperlink;

var image = new Image();

image.Width = 40;

image.Height = 40;

image.Opacity = 100;

image.Source = myImage;

//Добавление заведения на слой карты

\_dataLayer.Children.Add(image);

\_dataLayer.Children.Add(placeInfo);

var location = new Location(placeLatitude, placeLongitude);

var infoLocation = new Location(placeLatitude + 0.00012, placeLongitude + 0.0001);

MapLayer.SetPosition(placeInfo, infoLocation);

MapLayer.SetPosition(image, location);

MapLayer.SetPositionAnchor(image, new Point(image.Width / 2, image.Height / 2));

}

В событии ClickedHyperlink(object sender, RoutedEventArgs e) выполняется получение имени отправителя, т.е. имя гиперссылки, и переход на информационную страницу. Исходный код:

void ClickedHyperlink(object sender, RoutedEventArgs e)

{

//Получение имя отправителя(ID места)

Control control = (Control) sender;

var name = control.Name;

//Переход на информационную страницу и передача имени

this.Frame.Navigate(typeof(PlaceInfoPage), name);

}

Для получения сведений о месте необходимо выполнить запрос:

https://maps.googleapis.com/maps/api/place/details/output?parameters,где output?parameters должны содержать формат возвращаемого ответа, ключ Google API и ID места. Из ответа приложение получает такую информацию, как адрес, сайт, время работы и photo reference. Это реализовано в асинхронном методе IdentifyPlace():

private async void IdentifyPlace()

{

//Отправка запроса и получение ответа в формате XML

string baseUri = "https://maps.googleapis.com/maps/api/place/details/xml?placeid=" + \_placeId +

"&language=ru&key=AIzaSyC79KnpW4oNaZGZdYABoSrSP0fUqnKPe14";

var request = WebRequest.Create(baseUri);

var response = await request.GetResponseAsync();

var xDoc = XDocument.Load(response.GetResponseStream());

var status = xDoc.Element("PlaceDetailsResponse").Element("status");

if (status.Value == "OK")

{

//Анализ ответа

var place = xDoc.Element("PlaceDetailsResponse").Element("result");

\_placeName = place.Element("name").Value;

string address = place.Element("vicinity").Value;

string rating;

if (place.Elements("rating").Any())

{

rating = place.Element("rating").Value;

}

else

rating = "Нет оценок";

string webSite;

string []days = new string [7];

if (place.Elements("website").Any())

{

webSite = place.Element("website").Value;

}

else

webSite = "Не существует";

if (place.Elements("opening\_hours").Any())

{

IEnumerable<XElement> resultElement = from element in

place.Element("opening\_hours").Elements("weekday\_text")

select element;

int i = 0;

foreach (XElement element in resultElement)

{

days[i] = element.Value;

i++;

}

}

else days = null;

string photoReferense;

if (place.Elements("photo").Any())

{

photoReferense = place.Element("photo").Element("photo\_reference").Value;

}

else

photoReferense = "Не существует";

//Функция добавления фото

AddPlacePhoto(photoReferense);

//Функция добавления информации

AddPlaceInfo(\_placeName, address, rating, webSite, days);

//Получение меню из базы данных

await LoadLocalDataAsync();

}

else

{

//Ошибка, если не был получен ответ

MessageDialog msgDialog = new MessageDialog("Ошибка загрузки данных", "Ошибка");

UICommand okBtn = new UICommand("OK");

msgDialog.Commands.Add(okBtn);

await msgDialog.ShowAsync();

}

}

Чтобы получить фотографии места, нужно опять делать запрос. В этот раз запрос выглядит таким образом:

https://maps.googleapis.com /maps/api/place/photo?parameters, где photo?parameters должны содержать ключ Google API, максимальную высоту или ширину фото и photo reference**.** Для этого используется функция AddPlacePhoto(string photoReferense):

private void AddPlacePhoto(string photoReferense)

{

//Добавить фото, если оно существует

if (photoReferense != "Не существует")

{

string baseUri = "https://maps.googleapis.com/maps/api/place/photo?maxwidth=400&photoreference=" + photoReferense +

"&key=AIzaSyC79KnpW4oNaZGZdYABoSrSP0fUqnKPe14";

PlacePhoto.Source = new BitmapImage(new Uri(baseUri, UriKind.Absolute));

}

else

//Добавить фото "Не существует"

PlacePhoto.Source = new BitmapImage(new Uri("ms-appx:///Assets/no\_photo.jpg", UriKind.Absolute));

}

Меню ресторанов и кафе содержатся в базе данных приложения Café Guide. Элементы базы данных хранятся в формате JSON. Для импорта информации используется пространство имен Windows.Data.Json. Приложение не имеет базы данных всех ресторанов, т.к. написана только небольшая база данных для демонстрации работы приложения. База данных организована таким образом:

[

{

"McDonald's" : {"dishes" : ["Гранд Биг Мак",

"Цезарь Ролл",

"Биг Мак",

"Биг Тейсти",

"Чикенбургер",

"Чизбургер",

"Гамбургер"],

"prices" : [ 6.5,

5.95,

4.5,

7.7,

1.9,

1.9,

1.75

]

}

},

…

]

Т.е. в массиве содержатся объекты (название заведений), в которых содержится еще один объект, что облегчает поиск заведения в базе данных. Загрузка меню из базы данных выполняется в асинхронной задаче LoadLocalDataAsync():

public async Task LoadLocalDataAsync()

{

var file = await Package.Current.InstalledLocation.

GetFileAsync("Data\\Menu.txt");

var result = await FileIO.ReadTextAsync(file);

var menu = JsonArray.Parse(result);

CreateMenu(menu, \_placeName);

}

private void CreateMenu(JsonArray array, string placeName)

{

//Проверка на наличие заведения в базе данных

bool check = false;

foreach (var item in array)

{

var info = item.GetObject();

IJsonValue placeValue;

//Если есть название в базе данных, то записать

//в placeValue, если нет - продолжить цикл

if (!info.TryGetValue(placeName, out placeValue))

{

continue;

}

check = true;

var place = placeValue.GetObject();

foreach (var key in place.Keys)

{

IJsonValue value;

if (!place.TryGetValue(key, out value))

{

continue;

}

switch (key)

{

case "dishes":

var dishes = value.GetArray();

foreach (var dish in dishes)

{

var food = new TextBlock();

food.Text = dish.GetString();

TextBlockDesign(food);

DishStackPanel.Children.Add(food);

}

break;

case "prices":

var prices = value.GetArray();

foreach (var price in prices)

{

var food = new TextBlock();

food.Text = price.GetNumber().ToString();

food.HorizontalAlignment =

HorizontalAlignment.Center;

TextBlockDesign(food);

PriceStackPanel.Children.Add(food);

}

break;

}

}

}

if (!check)

{

//Вывести сообщение, если не найдено

var error = new TextBlock();

error.Text = "Нет информации";

TextBlockDesign(error);

error.HorizontalAlignment = HorizontalAlignment.Center;

error.Margin = new Thickness(0, 20, 0, 0);

MenuGrid.Children.Add(error);

Grid.SetRow(error, 1);

}

}

**4.2. Описание работы приложения**

При запуске приложения вызывается метод IdentifyMyCurrentLocation(), который определяет текущее положения пользователя. Этот метод также содержит вложенный метод FindTheNearestPlaces(), который определяет ближайшие кафе и рестораны. Таким образом, при записке приложения пользователю доступна информация о заведениях и ресторанах, находящихся от него в радиусе 1 км.

Как было сказано ранее, при поиске пользователя адреса, он будет перемещен на ту точку на карте, которая отражает содержимое запроса пользователя. Выполняется событие ClickedSearchAddress(object sender, RoutedEventArgs e), описанное в пункте 4.1.

При поиске пользователем заведения по названию происходит событие ClickedSearchPlace(object sender, RoutedEventArgs e). После выполнения события на карте остается только отметка о текущем положении и добавляется отметка о найденном заведении, в противном случае пользователь видит сообщение, что место не было найдено.

При нажатии кнопки «Текущее положение» опять вызывается метод IdentifyMyCurrentLocation(), в результате чего происходит то же самое, что и при запуске приложения.

При взаимодействии с названием заведения приложение переходит на информационную страницу. При этом в асинхронный метод IdentifyPlace() передается ID места, благодаря чему приложение узнает всю дополнительную информацию о заведении. В методе выполняются все вложенные в него методы, в результате чего вся информация появляется на экране пользователя.

**Заключение**

В результате работы над курсовым проектом было разработано приложение для поиска ближайших кафе и ресторанов.

В разработки были изучены язык разметки XAML, использование C#/XAML, Bing Maps SDK для написания Windows Store-приложений. Для этого были прочитаны книги, связанные с этой темой. Также была изучена соответствующая документация Microsoft.

Для использования The Google Places API Web Service и Google Maps Geocoding API была изучена их работа и прочитана документация Google на соответствующие темы.

Чтобы качественно работать с базами данных, были рассмотрены принципы работы с XML- и JSON документами. Также была самостоятельно написана база данных для демонстрации ее работы в приложении.

Таким образом, было создано приложение, благодаря которому пользователь может легко определить ближайшие кафе и рестораны, узнать необходимую ему о них информацию, что поможет ему сделать удачный выбор.

**Приложение 1. Язык разметки XAML**

XAML (eXtensible Application Markup Language) — расширяемый язык разметки приложений. Декларативный язык XAML основан на XML, но по сравнению с ним обладает дополнительной семантикой. Теги, атрибуты и значения, применяемые в XAML, однозначно соответствуют объектам, свойствам, событиям и другим элементам среды исполнения. Поэтому любую XAML-разметку можно выразить кодом на одном из языков программирования, например, на C#. Рассмотрим это на примере класса Foo. Для сравнения в листинге 1 приведен пример на языке C#, а в листинге 2 – на XAML.

***Листинг 1. Класс Foo***

namespace FooNamespace

{

public class Foo

{

public string SomeText { get; set; }

}

}

…

Foo foo = new Foo();

foo.SomeText = "Значение свойства";

…

***Листинг 2. Класс Foo в XAML разметке***

<FooNamespace:Foo SomeText = "Значение свойства"

xmlns:FooNamespace="using:FooNamespace"/>

Код <Foo SomeText = "Значение свойства"/> не будет работать без подключения нужного пространства имен. В XAML-разметке, как и в коде на языке C#, мы должны подключать пространства имен. В случае с C#-кодом пространства имен подключаются с помощью оператора using.

Допустим, класс FooUser использует класс Foo, но находится в другом пространстве имен. Если бы классы FooUser и Foo находились в одном пространстве имен, необходимость подключать его отсутствовала бы. В случае с XAML-разметкой подключать пространства имен нужно всегда. В XML для этого предусмотрен атрибут xmlns. В XAML-разметке пространства имен XML отображаются на пространства имен Windows Runtime с помощью специального синтаксиса. В листинге 2 для тега класса Foo и его дочерних элементов подключается пространство имен FooNamespace. При подключении пространства имен для него задается префикс, который в данном случае совпадает с именем пространства имен, но это необязательно.

Рассмотрим код страницы Windows Store-приложения (листинг 3)

***Листинг 3. Страница Windows Store-приложения***

<Page

x:Class="Cafe\_Guide.MainPage"

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

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

xmlns:local="using:Cafe\_Guide"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:bing="using:Bing.Maps"

mc:Ignorable="d" RequestedTheme="Light"

NavigationCacheMode="Required">

…

</Page>

Обратите внимание на пространство имен по умолчанию (листинг П1.7) и пространство имен "x". Это пространства имен XML, заданные ссылками. Среда исполнения автоматически соотносит системные пространства имен, заданные ссылками, с пространствами имен Windows Runtime. Для пространства имен по умолчанию префикс не указывается.

В пространстве имен по умолчанию определены такие элементы управления, как Grid, StackPanel и большинство других, входящих в стандартную поставку. Именно благодаря заданию пространства имен по умолчанию мы можем указывать стандартные элементы управления без какого-либо префикса.

В пространстве имен "x" определен атрибут имени (x:Name), ключа (x:Key), а также заданы другие вспомогательные значения и атрибуты (например, x:Null для значения NULL).

Для того чтобы удобно разместить элементы управления на странице приложения, требуются менеджеры размещения, основная задача которых — предложить нам разнообразные схемы разметки или компоновки других элементов управления. Наиболее распространены следующие менеджеры размещения:

* Canvas;
* StackPanel;
* Grid.

Менеджер размещения Canvas предоставляет наиболее простой вариант разметки. Он применяется для абсолютного позиционирования элементов управления по координатам. Для позиционирования элементов управления на Canvas служат прикрепленные свойства (Attached Properties), расширяющие свойства самих элементов управления. Разместим несколько кнопок (Button) на Canvas так, как показано в листинге 4.

***Листинг 4. Размещение кнопок на Canvas***

<Canvas>

<Button Canvas.Top="75" Canvas.Left="75" Content="Кнопка T75.L75.Z-1"

Canvas.ZIndex="-1" FontSize="18" Width="200"

Height="75" Background="Orange" />

<Button Canvas.Top="175" Canvas.Left="5" Content="Кнопка T175.L45"

FontSize="18" Width="200" Height="75" />

<Button Canvas.Top="110" Canvas.Left="90" Content="Кнопка T95.L90"

FontSize="18" Width="220" Height="75" />

</Canvas>

Менеджер размещения Canvas удобен, когда элементы управления внутри него не будут перемещаться, а окно приложения — менять ориентацию, или когда необходимо по тем или иным причинам позиционировать элементы интерфейса приложения в точности по заданным координатам. В противном случае использование Canvas может быть сложнее по сравнению с такими менеджерами размещения, как Grid или StackPanel. Поэтому менеджер размещения Canvas применяется относительно редко.

Менеджер размещения StackPanel предоставляет вариант разметки, который располагает помещенные в него элементы один за другим горизонтально или вертикально (по умолчанию — вертикально). Расположим несколько кнопок в StackPanel (листинг 5).

***Листинг 5. Размещение кнопок в StackPanel***

<StackPanel>

<Button Margin="0,0,0,10" Content="Кнопка 0.0.0.10"

FontSize="18" Width="200" Height="75" />

<Button Content="Кнопка 0.0.0.0" FontSize="18"

Width="200" Height="75" />

<Button Margin="0,50,0,0" Content="Кнопка 0.50.0.0"

FontSize="18" Width="200" Height="75" />

<Button Margin="150,170,0,0" Content="Кнопка 150.170.0.0"

FontSize="18" Width="200" Height="75" />

</StackPanel>

Менеджер размещения Grid позволяет позиционировать элементы внутри себя максимально гибко. Grid предоставляет возможность размещать элементы по строкам и столбцам. С помощью менеджера размещения Grid разработчик определяет общую структуру сетки, а потом, используя присоединенные свойства, размещает элементы управления в ячейках сетки.

Рассмотрим код из листинга 6.

***Листинг 6. Менеджер размещения Grid***

<Grid Width="600" Height="600">

<Grid.RowDefinitions>

<RowDefinition Height="200"/>

<RowDefinition Height="\*"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="200" />

</Grid.ColumnDefinitions>

<Button Grid.Column="0" Grid.Row="0" Content="Кнопка 0.0"

FontSize="18" Width="140" Height="75" />

<Button Grid.Column="2" Grid.Row="0" Content="Кнопка 2.0"

FontSize="18" Width="140" Height="75" />

<Button Grid.Column="1" Grid.Row="2" Content="Кнопка 1.2"

FontSize="18" Width="140" Height="75" />

</Grid>

При определении строк и столбцов могут быть реализованы все три возможных способа указания размера:

* прямое указание размера;
* автоматический размер, в зависимости от содержимого (Auto);
* пропорциональное разделение оставшегося пространства (\*).

Таким образом, в разметке мы имеем первую строку высотой 200 единиц, третью строку, высота которой подстраивается под содержимое, и вторую строку, занимающую все оставшееся место.

Если требуется разместить элементы управления сразу в нескольких строках, можно установить прикрепленное свойство Grid.RowSpan. Для размещения элемента управления в нескольких столбцах предназначено свойство Grid.ColumnSpan. В листинге 7 показана кнопка, занимающая две строки и три столбца.

***Листинг 7. Свойства Grid.RowSpan и Grid.ColumnSpan***

<Button Content="Кнопка"

VerticalAlignment="Stretch" HorizontalAlignment="Stretch"

Grid.RowSpan="2" Grid.ColumnSpan="3" />

**Список использованных источников**

1. С. Пугачев, А. Шериев, К. Кичинский «Разработка приложений для Windows 8 на языке C#», 2013

2. Сайт <https://en.wikipedia.org/wiki/Foursquare> - информация о Foursquare

3. Сайт <https://ru.wikipedia.org/wiki/Карты_Google> - информация о Google Maps

4. Bing Maps for Windows Store Apps. [Электронный ресурс] – Режим доступа:<https://msdn.microsoft.com/en-us/library/hh846481.aspx> - документация Microsoft об использовании Bing Maps для Windows Store-приложений

5. Detect location (XAML). [Электронный ресурс] – Режим доступа: https://msdn.microsoft.com/en-us/us-en/library/windows/apps/xaml/hh465120.aspx - документация Microsoft об определении местоположения

6. The Google Places API Web Service. [Электронный ресурс] – Режим доступа: <https://developers.google.com/places/web-service/> - документация Google об использовании Google Places API

7. Google Maps Geocoding API. [Электронный ресурс] – Режим доступа: https://developers.google.com/maps/documentation/geocoding/ - документация Google об использовании Google Maps Geocoding API

8. R. Brundritt «Location Intelligence for Windows Store Apps», 2014