**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Научный руководитель,  профессор департамента  программной инженерии  факультета компьютерных наук  канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.М. Авдошин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **АНДРОИД ПРИЛОЖЕНИЕ ПОИСКА БАКМОТАОВ,**  **ГДЕ МОЖНО СНЯТЬ НАЛИЧНЫЕ БЕЗ КОМИССИИ**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.16-01 12 01-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы БПИ193  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /В.А. Радайкин /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | |
|  | | |
|  | |  |

**Москва 2020**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU.17701729.04.16-01 12 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **АНДРОИД ПРИЛОЖЕНИЕ ПОИСКА БАКМОТАОВ,**  **ГДЕ МОЖНО СНЯТЬ НАЛИЧНЫЕ БЕЗ КОМИССИИ**  **Текст программы**  **RU.17701729.04.16-01 12 01-1**  **Листов 8** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**Москва 2020**

СОДЕРЖАНИЕ

1. ТЕКСТ ПРОГРАММЫ 3

1.1. Класс Atm.cs 3

1.2. Класс App.xaml.cs 3

1.3. Класс MainActivity.cs 3

1.4. Класс MainPage.xaml.cs 4

2. ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 10

# ТЕКСТ ПРОГРАММЫ

Программа написана на языке C# с использованием среды разработки Microsoft Vicual Studio. Программа состоит из 32 классов (учитываются только вручную написанные классы, полностью автоматически сгенерированные средой или движком классы не учитываются).

В данном документе содержится только вручную написанный код программы, код из полностью автоматически сгенерированных файлов в данном документе не представлен.

## Класс Atm.cs

class Atm

{

public string AtmName { get; set; }

public string Adress { get; set; }

public double Lat { get; set; }

public double Lng { get; set; }

}

## Класс App.xaml.cs

public partial class App : Application

{

public App()

{

InitializeComponent();

MainPage = new MainPage();

}

}

## Класс MainActivity.cs

[Activity(Label = "AtmMap", Icon = "@drawable/atm", Theme = "@style/MainTheme", MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize)]

public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity

{

const int RequestLocationId = 0;

readonly string[] LocationPermissions =

{

Manifest.Permission.AccessCoarseLocation,

Manifest.Permission.AccessFineLocation

};

protected override void OnCreate(Bundle savedInstanceState)

{

TabLayoutResource = Resource.Layout.Tabbar;

ToolbarResource = Resource.Layout.Toolbar;

base.OnCreate(savedInstanceState);

if ((int)Build.VERSION.SdkInt >= 23)

{

if (CheckSelfPermission(Manifest.Permission.AccessFineLocation) != Permission.Granted)

{

RequestPermissions(LocationPermissions, RequestLocationId);

}

}

Xamarin.Essentials.Platform.Init(this, savedInstanceState);

global::Xamarin.Forms.Forms.Init(this, savedInstanceState);

Xamarin.FormsMaps.Init(this, savedInstanceState);

LoadApplication(new App());

}

public override void OnRequestPermissionsResult(int requestCode, string[] permissions, [GeneratedEnum] Android.Content.PM.Permission[] grantResults)

{

Xamarin.Essentials.Platform.OnRequestPermissionsResult(requestCode, permissions, grantResults);

base.OnRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == RequestLocationId)

{

if ((grantResults.Length == 1) && (grantResults[0] == (int)Permission.Granted)) { }

}

else

{

base.OnRequestPermissionsResult(requestCode, permissions, grantResults);

}

}

}

## Класс MainPage.xaml.cs

public partial class MainPage : ContentPage

{

List<Atm> atmList; //В лист будет храниться список банкоматов, которые выбрал пользователь, это для того, чтобы повторно не получать xml и сократить время загрузки

Dictionary<string, List<string>> bankSponsorDictionary = new Dictionary<string, List<string>>() //Словарь партнёров

{

{"Тинькофф", new List<string>(){"юнистрим","сбербанк","втб","альфа-банк","райффайзенбанк", "газпромбанк", "росбанк", "промсвязьбанк", "уралсиб", "открытие", "Почта Банк" } },

{"Альфа банк", new List<string>(){"Уральский банк реконструкции и развития","Московский кредитный банк" }},

{"ВТБ", new List<string>(){"Почта Банк","Банк Возрождение", "Саровбизнесбанк","Запсибкомбанк" }},

{"Россельхоз", new List<string>(){"Альфа-банк","Райффайзенбанк", "Промсвязьбанк","Росбанк" }},

{"Почта банк", new List<string>(){"Фора-банк","ВТБ" }},

{"Промсвязьбанк",new List<string>(){"Промсвязьбанк","Газпромбанк" }},

{"Газпромбанк",new List<string>(){"Группа ВТБ","Почтабанк","Банк СГБ","МТС-Банк","Быстробанк" }}

};

public MainPage() =>

InitializeComponent();

private async void SelectBank\_Clicked(object sender, System.EventArgs e)

{

string nameBank = await DisplayActionSheet("Выберите банк", "Закрыть", "", "Сбербанк", "Тинькофф", "Альфа банк", "ВТБ", "Россельхоз", "Home Credit", "Почта банк", "Открытие", "Промсвязьбанк", "Газпромбанк"); //Окно выбора

if (nameBank != "Закрыть") //Проверяем, чтобы сработало только при выборе банка

{

map.Pins.Clear(); //чистим карту от старых меток

(sender as Button).Text = nameBank; //Применяем к тексту кнопки название банка

atmList = new List<Atm>();

map.Pins.Clear();

GetAtm(nameBank);

foreach (var item in bankSponsorDictionary)

{

if (item.Key == nameBank)

btnSelectTypeATM.IsEnabled = true;

}

SelectBank.IsEnabled = true;

btnSelectType.IsEnabled = true;

}

}

private async void GetAtm(string nameBank)

{

SelectBank.IsEnabled = false; //Блокируем кнопку, для того, чтобы во время загрузки не было лишних действий от пользователя

btnSelectType.IsEnabled = false; // Блокируем вторую кнопку

btnSelectTypeATM.IsEnabled = false;

XmlDocument xDoc = new XmlDocument(); //Создаём документ Xml, точнее класс для работы с документом

xDoc.Load($"https://maps.googleapis.com/maps/api/place/textsearch/xml?query=банкоматы+{nameBank}+в+Москве&key=AIzaSyBcLXggfCt9AnHPiGUq41mXKOKmeV4fy4E"); //Загружаем XML документ

XmlElement xRoot = xDoc.DocumentElement; // Получаем корневой элмент

foreach (XmlNode res in xRoot) //Обходим весь корневой элмент

{

Atm atm = new Atm(); // Создаём экземпляр нового банкомата

foreach (XmlNode result in res.ChildNodes) // обходим result, т.е один полученный банкомат

{

if (result.Name == "name") // получаем имя

atm.AtmName = result.InnerText;

if (result.Name == "formatted\_address") // получаем адресс

atm.Adress = result.InnerText;

}

atmList.Add(atm); // добавляем банкомат в список

}

foreach (Atm atm in atmList.ToList()) // проходим по списку

{

if (atm.AtmName != null) // это для того, чтобы не вылетало ошибки, т.к первая и посл запись null

{

var locations = await Geocoding.GetLocationsAsync(atm.Adress); //Получаем координаты по адресу

var location = locations?.FirstOrDefault();

atm.Lat = location.Latitude; //Приравниваем полученные координаты банкомата к банкомату в списке

atm.Lng = location.Longitude;

Pin pin = new Pin() //Создаем метку для карт

{

Label = atm.AtmName,

Address = atm.Adress,

Position = new Position(atm.Lat, atm.Lng)

};

map.Pins.Add(pin); // добавляем метку на карту

}

}

}

private async void ChangeTypeLook\_Clicked\_1(object sender, System.EventArgs e)

{

string typeLook = await DisplayActionSheet("Выберите банк", "Закрыть", "", "Все банкоматы", "Ближайший");

if (typeLook != "Закрыть")

{

btnSelectType.Text = typeLook;

btnSelectType.IsEnabled = false;

btnSelectTypeATM.IsEnabled = false;

switch (typeLook)

{

case "Все банкоматы":

foreach (Atm atm in atmList)

{

if (atm.AtmName != null)

{

Pin pin = new Pin()

{

Label = atm.AtmName,

Address = atm.Adress,

Position = new Position(atm.Lat, atm.Lng)

};

map.Pins.Add(pin);

}

}

break;

case "Ближайший":

Atm bestPositionAtm = new Atm();

double bestKilometers = 0;

var locationPeople = await Geolocation.GetLocationAsync(new GeolocationRequest(GeolocationAccuracy.Best));

Pin myLocation = new Pin()

{

Label = "Я",

Position = new Position(locationPeople.Latitude, locationPeople.Longitude)

};

foreach (Atm atm in atmList)

{

if (atm.AtmName != null)

{

double kilometrs = Location.CalculateDistance(new Location(locationPeople.Latitude, locationPeople.Longitude), new Location(atm.Lat, atm.Lng), DistanceUnits.Kilometers); //Высчитываем дистанцию

if (bestKilometers == 0)

{

bestKilometers = kilometrs;

bestPositionAtm.AtmName = atm.AtmName;

bestPositionAtm.Adress = atm.Adress;

bestPositionAtm.Lat = atm.Lat;

bestPositionAtm.Lng = atm.Lng;

}

else if (bestKilometers > kilometrs)

{

bestKilometers = kilometrs;

bestPositionAtm.AtmName = atm.AtmName;

bestPositionAtm.Adress = atm.Adress;

bestPositionAtm.Lat = atm.Lat;

bestPositionAtm.Lng = atm.Lng;

}

}

}

map.Pins.Clear();

Pin bestPin = new Pin()

{

Label = bestPositionAtm.AtmName,

Address = bestPositionAtm.Adress,

Position = new Position(bestPositionAtm.Lat, bestPositionAtm.Lng)

};

map.Pins.Add(myLocation);

map.Pins.Add(bestPin);

break;

}

btnSelectType.IsEnabled = true;

btnSelectTypeATM.IsEnabled = true;

}

}

private async void btnSelectTypeATM\_Clicked(object sender, System.EventArgs e)

{

string typeLook = await DisplayActionSheet("Выберите тип банкоматов", "Закрыть", "", "Все банкоматы", "Банкоматы банка", "Банкоматы партнёров");

if (typeLook != "Закрыть")

{

(sender as Button).Text = typeLook;

switch (typeLook)

{

case "Все банкоматы":

map.Pins.Clear();

atmList = new List<Atm>();

GetAtm(SelectBank.Text);

foreach (var item in bankSponsorDictionary)

{

if (item.Key == SelectBank.Text)

{

foreach (string result in item.Value)

{

GetAtm(result);

}

}

}

SelectBank.IsEnabled = true;

btnSelectType.IsEnabled = true;

btnSelectTypeATM.IsEnabled = true;

break;

case "Банкоматы банка":

map.Pins.Clear(); //чистим карту от старых меток

atmList = new List<Atm>();

map.Pins.Clear();

GetAtm(SelectBank.Text);

SelectBank.IsEnabled = true;

btnSelectType.IsEnabled = true;

break;

case "Банкоматы партнёров":

map.Pins.Clear();

atmList = new List<Atm>();

foreach (var item in bankSponsorDictionary)

{

if (item.Key == SelectBank.Text)

{

foreach (string result in item.Value)

{

GetAtm(result);

}

}

}

SelectBank.IsEnabled = true;

btnSelectType.IsEnabled = true;

btnSelectTypeATM.IsEnabled = true;

break;

}

}

}

}

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц)  в докум.) | №  документа | Входящий  №  сопроводит  ельного  докум. и дата. | Подп. | Дата |
| Изм. | Изменен. | Заменён. | Новых | Аннул. |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |