**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Руководитель департамента программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. М. Авдошин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |
| --- | --- |
| *Подп. и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подп. и дата* |  |
| *Инв. № подл* | *RU.17701729.04.01-01 12 01-1* |

**ПРОГРАММНЫЙ СИНТЕЗАТОР В ФОРМАТЕ ИНСТРУМЕНТА VST**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 12 01-1-ЛУ**

|  |  |
| --- | --- |
|  | Исполнитель:  студент группы БПИ182  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /М. А. Шакура /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

**2019**

|  |
| --- |
| УТВЕРЖДЕН  RU.17701729.04.01-01 12 01-1-ЛУ |

|  |  |
| --- | --- |
| *Подп. и дата* |  |
| *Инв. № дубл.* |  |
| *Взам. инв. №* |  |
| *Подп. и дата* |  |
| *Инв. № подл* | *RU.17701729.04.01-01 12 01-1* |

**ПРИЛОЖЕНИЕ АКТИВНОСТИ И УВЕДОИЛЕНИЯ ВШЭ**

**Текст программы**

**RU.17701729.04.01-01 12 01-1**

**Листов 20**

**2019**

# АННОТАЦИЯ

В данном программном документе представлен текст программы Приложение активности и уведомления ВШЭ». В разделе «Текст программы» находится текст программы, распределённый по файлам.

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 2](#_Toc8912161)

[1. ТЕКСТ ПРОГРАММЫ 4](#_Toc8912162)

[1.1. Models\AppData.cs 4](#_Toc8912163)

[1.2. Models\Item.cs 6](#_Toc8912164)

[1.3. Servises\IDataStore.cs 7](#_Toc8912165)

[1.4. Servises\MockDataStore.cs 7](#_Toc8912166)

[1.5. ViewModels\AboutViewModel.cs 8](#_Toc8912167)

[1.6. ViewModels\BaseViewModel.cs 8](#_Toc8912168)

[1.7. ViewModels\ItemDetailViewModel.cs 9](#_Toc8912169)

[1.8. ViewModels\ItemsViewModel.cs 9](#_Toc8912170)

[1.9. Views\AboutPage.xaml.cs 9](#_Toc8912171)

[1.10. Views\Favorite.xaml.cs 9](#_Toc8912172)

[1.11. Views\ItemDetailPage.xaml.cs 9](#_Toc8912173)

[1.12. Views\ItemsPage.xaml.cs 9](#_Toc8912174)

[1.13. Views\MainPage.xaml.cs 9](#_Toc8912175)

[1.14. App.xaml.cs 9](#_Toc8912176)

[1.15. Views\AboutPage.xaml 9](#_Toc8912177)

[1.16. Views\Favorite.xaml 9](#_Toc8912178)

[1.17. Views\ItemDetailPage.xaml 9](#_Toc8912179)

[1.18. Views\ItemsPage.xaml 9](#_Toc8912180)

[1.19. Views\MainPage.xaml 9](#_Toc8912181)

[1.20. App.xaml 9](#_Toc8912182)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 10](#_Toc8912183)

# ТЕКСТ ПРОГРАММЫ

## Models\AppData.cs

using System;

using System.Collections.Generic;

using Xamarin.Forms;

using System.IO;

using Newtonsoft.Json;

namespace App5.Models

{

/// <summary>

/// Sctruct of Data which writes to files

/// </summary>

struct FileStruct

{

public bool isrus { set; get; }

public bool IsThemeWhite { set; get; }

public List<string> Links;

}

static class AppData

{

static public Action SettingChanged;

static public List<string> Links = new List<string>();

/// <summary>

/// Langs

/// </summary>

static public string ru;

static public string en;

static public bool isrus { set; get; } = true;

/// <summary>

/// Colors

/// </summary>

static public bool IsThemeWhite { set; get; } = true;

static public Color BackgroundColor { get => IsThemeWhite ? Color.White : Color.FromRgb(16, 16, 16); }

static public Color BarBackgroundColor { get => IsThemeWhite ? Color.White : Color.FromRgb(10, 10, 10); }

static public Color FrontColor { get => IsThemeWhite ? Color.Black : Color.White; }

static AppData()

{

SettingChanged += Load;

/// <summary>

/// Read from file

/// </summary>

try

{

FileStruct a;

string json;

string path = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

string filePath = Path.Combine(path, "AppData.txt");

using (var file = File.Open(filePath, FileMode.Open, FileAccess.Read))

using (var strm = new StreamReader(file))

{

json = strm.ReadToEnd();

}

a = JsonConvert.DeserializeObject<FileStruct>(json);

isrus = a.isrus;

IsThemeWhite = a.IsThemeWhite;

Links = a.Links;

}

catch (Exception e){}

}

/// <summary>

/// Write to file

/// </summary>

static void Load()

{

try

{

FileStruct a = new FileStruct() {isrus=isrus,

IsThemeWhite= IsThemeWhite,

Links = Links};

string json = JsonConvert.SerializeObject(a);

string path = Environment.GetFolderPath(Environment.SpecialFolder.Personal);

string filePath = Path.Combine(path, "AppData.txt");

using (var file = File.Open(filePath, FileMode.Create, FileAccess.Write))

using (var strm = new StreamWriter(file))

{

strm.Write(json);

}

}

catch (Exception e){}

}

}

}

## Models\Item.cs

using Xamarin.Forms;

namespace App5.Models

{

public class Item

{

public string Type { get; set; }

public string Day { get; set; } = "";

public string Header { get; set; } = "";

public string Description { get => string.Join(", ", Tags); }

public string ExtraString { get => string.Join(", ", Extra); }

public string Link { get; set; } = "";

public string Place { get; set; } = "";

public string Time { get; set; } = "";

public string Date { get => Time == "" ? Day : Day + ", " + Time; }

public string[] Tags { set; get; } = { };

public string[] Extra { set; get; } = { };

public bool DayVisible { get => Type == "Day"; }

public bool ExtraVisible { get => ExtraString != "" && Type == "Event"; }

public bool DescriptionVisible { get => Description != "" && Type == "Event"; }

public bool TimeVisible { get => Time != "" && Type == "Event"; }

public bool PlaceVisible { get => Place != "" && Type == "Event"; }

public bool TagsVisible { get => Tags.Length > 0 && Type == "Event"; }

public bool HeaderVisible { get => Type == "Event"; }

public bool Clickable { get => Type == "Event"; }

public string FavSource { get => AppData.Links.Contains(Link) ? "Added.png" : "fplus.png"; }

public Color FrontColor { get => AppData.FrontColor;}

public Color BackgroundColor { get => AppData.BackgroundColor; }

}

}

## Servises\IDataStore.cs

using System.Collections.Generic;

using System.Threading.Tasks;

namespace App5.Services

{

public interface IDataStore<T>

{

Task<IEnumerable<T>> GetItemsAsync(bool forceRefresh = false);

}

}

## Servises\MockDataStore.cs

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Net;

using System.Threading.Tasks;

using App5.Models;

using Newtonsoft.Json;

using App5.Views;

namespace App5.Services

{

public class MockDataStore : IDataStore<Item>

{

/// <summary>

/// List of Items

/// </summary>

List<Item> items;

/// <summary>

/// Request to server

/// </summary>

/// <param name="url">Link</param>

/// <returns></returns>

static public string Get(string url)

{

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

request.AutomaticDecompression = DecompressionMethods.GZip | DecompressionMethods.Deflate;

try

{

using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())

using (Stream stream = response.GetResponseStream())

using (StreamReader reader = new StreamReader(stream))

return reader.ReadToEnd();

}

catch

{

return null;

}

}

/// <summary>

/// Reloadl Data from server

/// </summary>

static public void RealoadData()

{

AppData.ru = Get("https://shakura.dev/hseapi");

AppData.en = Get("https://shakura.dev/hseapien");

AboutPage.f();

}

/// <summary>

/// Local relaod data after swiching language

/// </summary>

public MockDataStore()

{

Item[] data = JsonConvert.DeserializeObject<Item[]>(AppData.isrus ? AppData.ru : AppData.en);

items = data.OfType<Item>().ToList();

}

public async Task<IEnumerable<Item>> GetItemsAsync(bool forceRefresh = false)

{

return await Task.FromResult(items);

}

}

}

## ViewModels\AboutViewModel.cs

using System.Windows.Input;

namespace App5.ViewModels

{

public class AboutViewModel : BaseViewModel

{

public ICommand OpenWebCommand { get; }

}

}

## ViewModels\BaseViewModel.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using Xamarin.Forms;

using App5.Models;

using App5.Services;

namespace App5.ViewModels

{

public class BaseViewModel : INotifyPropertyChanged

{

public IDataStore<Item> DataStore => DependencyService.Get<IDataStore<Item>>() ?? new MockDataStore();

bool isBusy = false;

public bool IsBusy

{

get { return isBusy; }

set { SetProperty(ref isBusy, value); }

}

string title = string.Empty;

public string Title

{

get { return title; }

set { SetProperty(ref title, value); }

}

protected bool SetProperty<T>(ref T backingStore, T value,

[CallerMemberName]string propertyName = "",

Action onChanged = null)

{

if (EqualityComparer<T>.Default.Equals(backingStore, value))

return false;

backingStore = value;

onChanged?.Invoke();

OnPropertyChanged(propertyName);

return true;

}

#region INotifyPropertyChanged

public event PropertyChangedEventHandler PropertyChanged;

protected void OnPropertyChanged([CallerMemberName] string propertyName = "")

{

var changed = PropertyChanged;

if (changed == null)

return;

changed.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

#endregion

}

}

## ViewModels\ItemDetailViewModel.cs

using System;

using App5.Models;

namespace App5.ViewModels

{

public class ItemDetailViewModel : BaseViewModel

{

public Item Item { get; set; }

public ItemDetailViewModel(Item item = null)

{

Title = item?.Header;

Item = item;

}

}

}

## ViewModels\ItemsViewModel.cs

using System;

using System.Collections.ObjectModel;

using System.Threading.Tasks;

using Xamarin.Forms;

using App5.Models;

using App5.Views;

namespace App5.ViewModels

{

public class Command1 : Command

{

public Command1(Action execute) : base(execute) { }

public void Execute(object sender, EventArgs args)

=> Execute(null);

}

public class ItemsViewModel : BaseViewModel

{

public ObservableCollection<Item> Items { get; set; } = new ObservableCollection<Item>();

public ObservableCollection<Item> FavItems { get; set; } = new ObservableCollection<Item>();

static System.Collections.Generic.IEnumerable<Item> items;

public Command LoadItemsCommand { get; set; }

static public Command LoadItemsCommand1 { get; set; }

static public Command1 SearchItems { get; set; }

static public Command1 SearchFavotitesItems { get; set; }

public ItemsViewModel()

{

LoadItemsCommand = new Command(async () => await ExecuteLoadItemsCommand());

LoadItemsCommand1 = LoadItemsCommand;

SearchItems = new Command1(async () => await ExecuteSearchItems());

SearchFavotitesItems = new Command1(async () => await ExecuteSearchFavotitesItems());

}

/// <summary>

/// Laod Items

/// </summary>

/// <returns></returns>

async Task ExecuteLoadItemsCommand()

{

if (IsBusy)

return;

IsBusy = true;

Items.Clear();

FavItems.Clear();

items = await DataStore.GetItemsAsync(true);

string Day = "";

string Day1 = "";

foreach (var item in items)

{

if (item.Day != Day)

Items.Add(new Item() { Type = "Day", Day = Day = item.Day });

Items.Add(item);

if (AppData.Links.Contains(item.Link))

{

if (item.Day != Day1)

FavItems.Add(new Item() { Type = "Day", Day = Day1 = item.Day });

FavItems.Add(item);

}

}

IsBusy = false;

}

/// <summary>

/// Searching in list of items

/// </summary>

/// <returns></returns>

async Task ExecuteSearchItems()

{

if (IsBusy)

return;

IsBusy = true;

string SearchQuery = ItemsPage.SearchQuery;

Items.Clear();

string Day = "";

foreach (var item in items)

{

if (item.Header.ToLower().

Contains(SearchQuery.ToLower()) ||

item.Description.ToLower().

Contains(SearchQuery.ToLower()) ||

item.Place.ToLower().

Contains(SearchQuery.ToLower())||

item.Date.ToLower().

Contains(SearchQuery.ToLower()))

{

if (item.Day != Day)

{

Items.Add(new Item() { Type = "Day", Day = Day = item.Day });

}

Items.Add(item);

}

}

IsBusy = false;

}

/// <summary>

/// Searching in Favorits list of items

/// </summary>

/// <returns></returns>

async Task ExecuteSearchFavotitesItems()

{

if (IsBusy)

return;

IsBusy = true;

string SearchQuery = Favorite.SearchQuery;

FavItems.Clear();

string Day = "";

foreach (var item in items)

{

if (AppData.Links.Contains(item.Link) && (item.Header.ToLower().

Contains(SearchQuery.ToLower()) ||

item.Description.ToLower().

Contains(SearchQuery.ToLower()) ||

item.Place.ToLower().

Contains(SearchQuery.ToLower()) ||

item.Date.ToLower().

Contains(SearchQuery.ToLower())))

{

if (item.Day != Day)

{

FavItems.Add(new Item() { Type = "Day", Day = Day = item.Day });

}

FavItems.Add(item);

}

}

IsBusy = false;

}

}

}

## Views\AboutPage.xaml.cs

using System;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using App5.Models;

using App5.ViewModels;

namespace App5.Views

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class AboutPage : ContentPage

{

public static event Action ThemeChanged;

public static void f() => ThemeChanged();

public AboutPage()

{

InitializeComponent();

ThemeChanged += UIUpdate;

English.IsEnabled = AppData.isrus;

Russian.IsEnabled = !AppData.isrus;

}

/// <summary>

/// lang changed

/// </summary>

/// <param name="sender"></param>

/// <param name="args"></param>

async void Russian\_Clicked(object sender, EventArgs args)

{

Russian.IsEnabled = false;

English.IsEnabled = true;

AppData.isrus = true;

AppData.SettingChanged();

ItemsViewModel.LoadItemsCommand1.Execute(null);

}

/// <summary>

/// lang changed

/// </summary>

/// <param name="sender"></param>

/// <param name="args"></param>

async void English\_Clicked(object sender, EventArgs args)

{

English.IsEnabled = false;

Russian.IsEnabled = true;

AppData.isrus = false;

AppData.SettingChanged();

ItemsViewModel.LoadItemsCommand1.Execute(null);

}

/// <summary>

/// theme changed

/// </summary>

/// <param name="sender"></param>

/// <param name="args"></param>

async void Black\_Clicked(object sender, EventArgs args)

{

AppData.IsThemeWhite = false;

AppData.SettingChanged();

ThemeChanged();

}

/// <summary>

/// theme changed

/// </summary>

/// <param name="sender"></param>

/// <param name="args"></param>

async void White\_Clicked(object sender, EventArgs args)

{

AppData.IsThemeWhite = true;

AppData.SettingChanged();

ThemeChanged();

}

void UIUpdate()

{

StackLayout1.BackgroundColor = AppData.BackgroundColor;

LANG\_TEXT.TextColor = AppData.FrontColor;

THEME\_TEXT.TextColor = AppData.FrontColor;

}

}

}

## Views\Favorite.xaml.cs

using System;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using App5.Models;

using App5.ViewModels;

namespace App5.Views

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class Favorite : ContentPage

{

ItemsViewModel viewModel;

/// <summary>

/// search bar query

/// </summary>

static public string SearchQuery = "";

void SearchQueryUpdate(object sender, EventArgs e)

{

SearchQuery = SearchBar.Text;

}

public Favorite()

{

InitializeComponent();

BindingContext = viewModel = new ItemsViewModel();

SearchBar.TextChanged += SearchQueryUpdate;

SearchBar.TextChanged += ItemsViewModel.SearchFavotitesItems.Execute;

AboutPage.ThemeChanged += UIUpdate;

}

async void OnItemSelected(object sender, SelectedItemChangedEventArgs args)

{

var item = args.SelectedItem as Item;

if (item == null)

return;

await Navigation.PushAsync(new ItemDetailPage(new ItemDetailViewModel(item)));

ItemsListView1.SelectedItem = null;

}

/// <summary>

/// search bar clicked

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

async void Search\_Clicked(object sender, EventArgs e)

{

SearchBar.IsVisible = !SearchBar.IsVisible;

Header.Icon = SearchBar.IsVisible ? "@drawable/cancel.png" : "@drawable/search.png";

if (SearchBar.IsVisible)

SearchBar.Focus();

else

SearchBar.Text = "";

}

/// <summary>

/// interface update

/// </summary>

void UIUpdate()

{

viewModel.LoadItemsCommand.Execute(null);

if (SearchBar.IsVisible)

{

string a = SearchBar.Text;

SearchBar.Text = "";

SearchBar.Text = a;

}

}

}

}

## Views\ItemDetailPage.xaml.cs

using System;

using System.Threading;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using App5.Models;

using App5.ViewModels;

namespace App5.Views

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ItemDetailPage : ContentPage

{

ItemDetailViewModel viewModel;

public ItemDetailPage(ItemDetailViewModel viewModel)

{

InitializeComponent();

BindingContext = this.viewModel = viewModel;

UIUpdate();

AboutPage.ThemeChanged += UIUpdate;

}

public void OpenLink()

{

string s = viewModel.Item.Link;

s = s[0] == '/' ? "https://hse.ru" + s : s;

Device.OpenUri(new Uri(s));

}

async void FavClicked(object sender, EventArgs args)

{

if (AppData.Links.Contains(viewModel.Item.Link))

AppData.Links.Remove(viewModel.Item.Link);

else

AppData.Links.Add(viewModel.Item.Link);

Header.Icon = viewModel.Item.FavSource;

ItemsViewModel.LoadItemsCommand1.Execute(null);

AppData.SettingChanged();

}

async void OnButtonClicked(object sender, EventArgs args)

{

var th = new Thread(OpenLink);

th.Start();

}

/// <summary>

/// interface update

/// </summary>

void UIUpdate()

{

UIHeader.TextColor = AppData.FrontColor;

UIExtraString.TextColor = AppData.FrontColor;

UIDate.TextColor = AppData.FrontColor;

UIPlace.TextColor = AppData.FrontColor;

}

}

}

## Views\ItemsPage.xaml.cs

using System;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using App5.Models;

using App5.ViewModels;

using App5.Services;

namespace App5.Views

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class ItemsPage : ContentPage

{

static public ItemsViewModel viewModel;

/// <summary>

/// search bar query

/// </summary>

static public string SearchQuery = "";

void SearchQueryUpdate(object sender, EventArgs e)

{

SearchQuery = SearchBar.Text;

}

void reload(object sender, EventArgs e)

{

MockDataStore.RealoadData();

}

public ItemsPage()

{

InitializeComponent();

BindingContext = viewModel = new ItemsViewModel();

SearchBar.TextChanged += SearchQueryUpdate;

SearchBar.TextChanged += ItemsViewModel.SearchItems.Execute;

ItemsListView1.Refreshing += reload;

AboutPage.ThemeChanged += UIUpdate;

}

async void OnItemSelected(object sender, SelectedItemChangedEventArgs args)

{

var item = args.SelectedItem as Item;

if (item == null)

return;

await Navigation.PushAsync(new ItemDetailPage(new ItemDetailViewModel(item)));

ItemsListView1.SelectedItem = null;

}

/// <summary>

/// search bar clicked

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

async void Search\_Clicked(object sender, EventArgs e)

{

SearchBar.IsVisible = !SearchBar.IsVisible;

Header.Icon = SearchBar.IsVisible ? "@drawable/cancel.png" : "@drawable/search.png";

if (SearchBar.IsVisible)

SearchBar.Focus();

else

SearchBar.Text = "";

}

bool lang = AppData.isrus;

protected override void OnAppearing()

{

base.OnAppearing();

if (viewModel.Items.Count == 0 || lang!= AppData.isrus)

viewModel.LoadItemsCommand.Execute(null);

lang = AppData.isrus;

}

/// <summary>

/// interface update

/// </summary>

void UIUpdate()

{

viewModel.LoadItemsCommand.Execute(null);

if (SearchBar.IsVisible)

{

string a = SearchBar.Text;

SearchBar.Text = "";

SearchBar.Text = a;

}

}

}

}

## Views\MainPage.xaml.cs

using System;

using App5.Models;

using App5.ViewModels;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace App5.Views

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MainPage : TabbedPage

{

public MainPage()

{

InitializeComponent();

AboutPage.ThemeChanged += UIUpdate;

}

void UIUpdate()

{

ToolBar.BarBackgroundColor = AppData.BarBackgroundColor;

ToolBar.BarTextColor = AppData.FrontColor;

BackgroundColor = AppData.BackgroundColor;

Xamarin.Forms.PlatformConfiguration.AndroidSpecific.TabbedPage.SetBarItemColor(ToolBar, AppData.FrontColor);

Xamarin.Forms.PlatformConfiguration.AndroidSpecific.TabbedPage.SetBarSelectedItemColor(ToolBar, Color.FromRgb(93, 188, 210));

}

}

}

## App.xaml.cs

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using App5.Views;

using App5.Services;

[assembly: XamlCompilation(XamlCompilationOptions.Compile)]

namespace App5

{

public partial class App : Application

{

public App()

{

InitializeComponent();

MainPage = new MainPage();

}

protected override void OnStart()

{

MockDataStore.RealoadData();

}

protected override void OnSleep()

{

}

protected override void OnResume()

{

}

}

}

## Views\AboutPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="App5.Views.AboutPage"

xmlns:vm="clr-namespace:App5.ViewModels"

Title="Настройки"

x:Name="ContentPage1">

<StackLayout Padding="10"

BackgroundColor="White"

x:Name="StackLayout1">

<Grid >

<Label Text="Язык: "

x:Name="LANG\_TEXT"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemTextStyle}"

FontSize="Large"

XAlign="Start"

TextColor="Black"/>

<Button Text="Русский"

x:Name="Russian"

Clicked="Russian\_Clicked"

BackgroundColor="#1658DA"

IsEnabled="False"

TextColor="White"

Margin="0, 0, 80, 0"

HorizontalOptions="End"/>

<Button Text="English"

x:Name="English"

Clicked="English\_Clicked"

BackgroundColor="#1658DA"

TextColor="White"

HorizontalOptions="End"

IsEnabled="True"/>

</Grid>

<StackLayout

Orientation="Vertical">

<BoxView

VerticalOptions="CenterAndExpand"

HorizontalOptions="Fill"

HeightRequest="1"

Color="#888888"/>

</StackLayout>

<Grid >

<Label Text="Тема: "

x:Name="THEME\_TEXT"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemTextStyle}"

FontSize="Large"

XAlign="Start"

TextColor="Black"/>

<Button Text="День"

x:Name="Day"

TextColor="Black"

Margin="0, 0, 80, 0"

BackgroundColor="#f7f7f7"

HorizontalOptions="End"

Clicked="White\_Clicked"/>

<Button Text="Ночь"

x:Name="Night"

TextColor="White"

BackgroundColor="#262626"

HorizontalOptions="End"

Clicked="Black\_Clicked"/>

</Grid>

</StackLayout>

</ContentPage>

## Views\Favorite.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="App5.Views.Favorite"

Title="Избранное">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="Header" Icon="@drawable/search.png" Clicked="Search\_Clicked" />

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<SearchBar x:Name="SearchBar"

VerticalOptions="Start"

BackgroundColor="#1658DA"

TextColor="White"

CancelButtonColor="White"

PlaceholderColor="#2d2d2d"

Placeholder="Поиск"

IsVisible="False"/>

<ListView x:Name="ItemsListView1"

ItemsSource="{Binding FavItems}"

VerticalOptions="FillAndExpand"

HasUnevenRows="true"

RefreshCommand="{Binding LoadItemsCommand}"

IsPullToRefreshEnabled="False"

IsRefreshing="{Binding IsBusy, Mode=OneWay}"

CachingStrategy="RecycleElement"

ItemSelected="OnItemSelected"

SeparatorColor="Gray">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell IsEnabled="{Binding Clickable}">

<StackLayout Padding="10" BackgroundColor="{Binding BackgroundColor}">

<Grid IsVisible="{Binding DayVisible}">

<Image Source="calendarlist.png"

WidthRequest="32"

HorizontalOptions="Start"/>

<Label Text="{Binding Day}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemTextStyle}"

FontSize="Large"

Margin="42, 0, 0, 0"

XAlign="Center"

TextColor="{Binding FrontColor}"/>

</Grid>

<Label Text="{Binding Time}"

IsVisible="{Binding TimeVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemDetailTextStyle}"

FontSize="16"

TextColor="#777777"/>

<Label Text="{Binding Header}"

x:Name="UIHeader"

IsVisible="{Binding HeaderVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemTextStyle}"

FontSize="Medium"

TextColor="{Binding FrontColor}"/>

<Label Text="{Binding Place}"

IsVisible="{Binding PlaceVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemDetailTextStyle}"

FontSize="16"

TextColor="#777777"/>

<Label Text="{Binding Description}"

IsVisible="{Binding TagsVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemDetailTextStyle}"

FontSize="16"

TextColor="#007AC5"/>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

## Views\ItemDetailPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="App5.Views.ItemDetailPage"

Title="Мероприятие">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="Header" Icon="{Binding Item.FavSource}" Clicked="FavClicked" />

</ContentPage.ToolbarItems>

<StackLayout Spacing="20" Padding="15">

<Label Text="{Binding Item.Header}" x:Name="UIHeader" FontSize="Large" TextColor="Black"/>

<Label Text="{Binding Item.ExtraString}" x:Name="UIExtraString" IsVisible="{Binding Item.ExtraVisible}" FontSize="Medium" XAlign="End"/>

<Label Text="{Binding Item.Date}" x:Name="UIDate" FontSize="Medium"/>

<Label Text="{Binding Item.Place}" x:Name="UIPlace" IsVisible="{Binding Item.PlaceVisible}" FontSize="Medium"/>

<Label Text="{Binding Item.Description}" x:Name="UIDescription" IsVisible="{Binding Item.DescriptionVisible}" FontSize="Medium" TextColor="#007AC5"/>

<Button Text="Открыть страницу мероприятия"

BackgroundColor="#1658DA"

TextColor="White"

Clicked="OnButtonClicked"/>

</StackLayout>

</ContentPage>

## Views\ItemsPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="App5.Views.ItemsPage"

Title="Собыитя">

<ContentPage.ToolbarItems>

<ToolbarItem x:Name="Header" Icon="@drawable/search.png" Clicked="Search\_Clicked" />

</ContentPage.ToolbarItems>

<ContentPage.Content>

<StackLayout>

<SearchBar x:Name="SearchBar"

VerticalOptions="Start"

BackgroundColor="#1658DA"

TextColor="White"

CancelButtonColor="White"

PlaceholderColor="#2d2d2d"

Placeholder="Поиск"

IsVisible="False"></SearchBar>

<ListView x:Name="ItemsListView1"

ItemsSource="{Binding Items}"

VerticalOptions="FillAndExpand"

HasUnevenRows="true"

RefreshCommand="{Binding LoadItemsCommand}"

IsPullToRefreshEnabled="True"

IsRefreshing="{Binding IsBusy, Mode=OneWay}"

CachingStrategy="RecycleElement"

ItemSelected="OnItemSelected"

SeparatorColor="Gray">

<ListView.ItemTemplate>

<DataTemplate>

<ViewCell IsEnabled="{Binding Clickable}">

<StackLayout Padding="10" BackgroundColor="{Binding BackgroundColor}">

<Grid IsVisible="{Binding DayVisible}">

<Image Source="calendarlist.png"

WidthRequest="32"

HorizontalOptions="Start"/>

<Label Text="{Binding Day}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemTextStyle}"

FontSize="Large"

Margin="42, 0, 0, 0"

XAlign="Center"

TextColor="{Binding FrontColor}"/>

</Grid>

<Label Text="{Binding Time}"

IsVisible="{Binding TimeVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemDetailTextStyle}"

FontSize="16"

TextColor="#777777"/>

<Label Text="{Binding Header}"

IsVisible="{Binding HeaderVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemTextStyle}"

FontSize="Medium"

TextColor="{Binding FrontColor}"/>

<Label Text="{Binding Place}"

IsVisible="{Binding PlaceVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemDetailTextStyle}"

FontSize="16"

TextColor="#777777"/>

<Label Text="{Binding Description}"

IsVisible="{Binding TagsVisible}"

LineBreakMode="WordWrap"

Style="{DynamicResource ListItemDetailTextStyle}"

FontSize="16"

TextColor="#007AC5"/>

</StackLayout>

</ViewCell>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</StackLayout>

</ContentPage.Content>

</ContentPage>

## Views\MainPage.xaml

<?xml version="1.0" encoding="utf-8" ?>

<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:android="clr-namespace:Xamarin.Forms.PlatformConfiguration.AndroidSpecific;assembly=Xamarin.Forms.Core"

android:TabbedPage.ToolbarPlacement="Bottom"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:views="clr-namespace:App5.Views"

x:Class="App5.Views.MainPage"

BarBackgroundColor="White"

x:Name="ToolBar"

>

<TabbedPage.Children >

<NavigationPage x:Name="NavigationPage1" >

<NavigationPage.Icon >

<OnPlatform x:TypeArguments="FileImageSource" x:Name="OnPlatform1" >

<On Platform="Android " Value="@drawable/calendar.png" x:Name="CalendarIcon"/>

</OnPlatform>

</NavigationPage.Icon>

<x:Arguments>

<views:ItemsPage />

</x:Arguments>

</NavigationPage>

<NavigationPage >

<NavigationPage.Icon>

<OnPlatform x:TypeArguments="FileImageSource">

<On Platform="Android " Value="@drawable/favourite.png"/>

</OnPlatform>

</NavigationPage.Icon>

<x:Arguments>

<views:Favorite />

</x:Arguments>

</NavigationPage>

<NavigationPage >

<NavigationPage.Icon>

<OnPlatform x:TypeArguments="FileImageSource">

<On Platform="Android " Value="@drawable/menu.png"/>

</OnPlatform>

</NavigationPage.Icon>

<x:Arguments>

<views:AboutPage />

</x:Arguments>

</NavigationPage>

</TabbedPage.Children>

</TabbedPage>

## App.xaml

<?xml version="1.0" encoding="utf-8" ?>

<Application xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="App5.App">

<Application.Resources>

<ResourceDictionary>

<!--Global Styles-->

<Color x:Key="NavigationPrimary">#1658DA</Color>

<Style TargetType="NavigationPage">

<Setter Property="BarBackgroundColor" Value="#1658DA"/>

<Setter Property="BarTextColor" Value="White" />

</Style>

</ResourceDictionary>

</Application.Resources>

</Application>

## hseapi.py

from flask import json

def hsepars(a):

Events1 = a.findAll("div", {"class": "b-events"})

Announcements = []

for e1 in Events1:

Day = e1.findAll("div", {"class": "b-events\_\_title"})[0].getText()

# Announcements.append({"Type": "Day", "Day": Day})

Events = e1.findAll("div", {"class": "b-events\_\_item js-events-item"})

for e in Events:

Header = e.findAll("div", {"class": "b-events\_\_body\_title large"})[0].getText()

Link = e.findAll("div", {"class": "b-events\_\_body\_title large"})[0].a['href']

PreTags = e.findAll("div", {"class": "tag-set smaller"})[0]

PreExtra = e.findAll("span", {"class": "grey small"})

Time = e.findAll("div", {"class": "b-events\_\_extra date"})[0].getText()

Tags = []

Extra = []

for tag in PreTags:

Tags.append(tag.getText())

for extra in PreExtra:

Extra.append(extra.getText())

Place = ""

try:

Place = e.findAll("p", {"class": "b-text-grey small"})[0].getText()

except:

pass

Announcements.append(

{"Type": "Event", "Header": Header, "Tags": Tags, "Link": Link, "Place": Place, "Extra": Extra,

"Time": Time, "Day": Day})

return json.dumps(Announcements)

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |