**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. №подл |  |

СОГЛАСОВАНО

Преподаватель департамента программной инженерии факультета компьютерных наук

А.Н. Степанов

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

УТВЕРЖДАЮ

Академический руководитель образовательной программы «Программная инженерия», к.т.н., профессор ДПИ ФКН

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**МОБИЛЬНАЯ ОНЛАЙН БИБЛИОТЕКА ПОСВЯЩЁННАЯ ВОЕННОЙ ИСТОРИИ СРЕДНИХ ВЕКОВ И РЕНЕССАНСА**

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.05-01 12 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ1910

\_\_\_\_\_\_\_\_\_\_\_\_ / А.Е. Семичев /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**Москва 2020**

**УТВЕРЖДЕН**

**RU.17701729.04.05-01 12 01-1 ЛУ**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. №подл |  |

**МОБИЛЬНАЯ ОНЛАЙН БИБЛИОТЕКА ПОСВЯЩЁННАЯ ВОЕННОЙ ИСТОРИИ СРЕДНИХ ВЕКОВ И РЕНЕССАНСА**

**Текст программы**

**RU.17701729.04.05-01 12 01-1**

**Листов 28**

# Аннотация

В данном программном документе приведет текст приложения «Мобильная онлайн библиотека посвящённая военной истории средних веков и ренессанса». Исходным языком данной разработки являются С#. Среда разработки – Visual Studio 2019.

**Москва 2020**

[Аннотация 2](#_Toc40655655)

[ТЕКСТ ПРОГРАММЫ 3](#_Toc40655656)

[1. Решение IronLib 4](#_Toc40655657)

[1.1. Библиотека классов “ClassLib” 4](#_Toc40655658)

[1.1.1. Перечисление TypeOfContent 4](#_Toc40655659)

[1.1.3. Класс MyPageInMain 5](#_Toc40655660)

[1.1.4. Класс MyPageInContent 5](#_Toc40655661)

[1.2. Решение Xamarin IronLib 6](#_Toc40655662)

[1.2.1. MainPage 6](#_Toc40655663)

[1.2.1.1. MainPage.xaml 6](#_Toc40655664)

[1.2.1.2. Класс MainPage.xaml.cs 7](#_Toc40655665)

[1.2.2. PageWithText 10](#_Toc40655666)

[1.2.2.1. PageWithText.xaml 10](#_Toc40655667)

[1.2.2.2. PageWithText.xaml.cs 11](#_Toc40655668)

[2. Решение APIForDB 12](#_Toc40655669)

[2.1. Класс Startup 12](#_Toc40655670)

[2.2. Класс DBController.cs. Папка Controllers 14](#_Toc40655671)

[2.3. Класс DBService.cs. Папка Services 16](#_Toc40655672)

[2.4. Класс DBRepository.cs. Папка Repositories 18](#_Toc40655673)

[2.6.1. Папка DB 24](#_Toc40655674)

[2.6.1.1. Класс AddPageRequest 24](#_Toc40655675)

[2.6.1.2. Класс AddPageResponse 25](#_Toc40655676)

[2.6.1.3. Класс GetDBResponse 25](#_Toc40655677)

[2.6.1.4. Класс Page 25](#_Toc40655678)

[2.6.1.5. Класс UpdatePageRequest 26](#_Toc40655679)

[2.6.1.6. Класс UpdatePageResponse 26](#_Toc40655680)

[2.6.2. Класс DbConnectionOption. Папка Options 26](#_Toc40655681)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 27](#_Toc40655682)

[ПРИЛОЖЕНИЕ 1 3](#_Toc40655683)

# ТЕКСТ ПРОГРАММЫ

Программа состоит из 2 решений одна относится к программе на xamarin, где будет приведён как xaml, так и под C#. Также будет включена библиотека классов. Второе решение реализация БД и API, что этой БД руководит. Деле эти решения и их реализации

# Решение IronLib

## Библиотека классов “ClassLib”

## Перечисление TypeOfContent

*/// <summary>  
/// перечесления типов для класса-конвертера  
/// </summary>*public enum TypeOfContent  
{  
 Label,

Image

}

* + 1. Класс MyContent

///<summary>  
/// класс-конвертер из моих классов в классы View  
/// </summary>

*//класс, через который работает серриализация ВСЕГО, что я получаю из БД. Изначально я хотел серриализовать List<View>, но это*

*1) не эффективно 2)не работает*

using System;  
using Xamarin.Forms;[Serializable]

public class MyContent

{

//Text, FontAttri, Size для переделки в Label

public string Text { get; set; }

public FontAttributes FontAttri { get; set; }

public NamedSize Size { get; set; }

/// <summary>

/// ссылка на изображение

/// </summary>

public string UrlToPict { get; set; }

/// <summary>

/// для определения типа в методе Add

/// </summary>

public TypeOfContent Type { get; set; }

/// <summary>

/// метод конвертирующий из моего класса в View

/// </summary>

/// <returns></returns>

public View Add()

{

if (Type == TypeOfContent.Label)

{

return new Label { Text = Text, FontAttributes = FontAttri, FontSize = Device.GetNamedSize(Size, typeof(Label)) };

}

else

{

return new Image { Source = new UriImageSource { Uri = new Uri($"{UrlToPict}") } };

}

}

}

## Класс MyPageInMain

using System;  
/// <summary>

/// класс, который содержит информацию для ячейки для главного меню

/// </summary>

[Serializable]

public class MyPageInMain

{

/// <summary>

/// изображение, которым помеченна статья на главной

/// </summary>

public MyContent ContentsForMainImage { get; set; }

/// <summary>

/// название статьи

/// </summary>

public MyContent ContentsForMainLabel { get; set; }

}

## Класс MyPageInContent

using System;

using System.Collections.Generic;

using Xamarin.Forms;  
/// <summary>

/// класс, который содержит информацию статьи. т.е. текст, ссылки на изображения

/// </summary>

[Serializable]

public class MyPageInContent

{

public MyPageInContent()

{

}

public MyPageInContent(List<MyContent> contents)

{

Contents = contents;

}

/// <summary>

/// контент статьи

/// </summary>

public List<MyContent> Contents { get; set; }

/// <summary>

/// Вывод всего содержимого листа с конверцией в класс View

/// </summary>

/// <param name="stackLayout"></param>

public void AddAll(StackLayout stackLayout)

{

foreach (var obj in Contents)

{

stackLayout.Children.Add(obj.Add());

}

}

}

## Решение Xamarin IronLib

## MainPage

## MainPage.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"

xmlns:d=<http://xamarin.com/schemas/2014/forms/design>

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

mc:Ignorable="d"

x:Class="IronLib.MainPage">

<ScrollView >

<Grid x:Name="grid" Margin="10,10,10,10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="0.5\*"/>

<ColumnDefinition Width="0.5\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="150"/>

<RowDefinition Height="150"/>

<RowDefinition Height="150"/>

<RowDefinition Height="150"/>

<RowDefinition Height="150"/>

</Grid.RowDefinitions>

</Grid>

</ScrollView>

</ContentPage>

## Класс MainPage.xaml.cs

using ClassLib;

using Newtonsoft.Json.Linq;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Net.Http;

using Xamarin.Forms;

namespace IronLib

{

[DesignTimeVisible(false)]

public partial class MainPage : ContentPage

{

/// <summary>

/// переменная для запросов к БД

/// </summary>

static readonly HttpClient client = new HttpClient();

/// <summary>

/// для выбора случайных статей

/// </summary>

static readonly Random rnd = new Random();

public MainPage()

{

InitializeComponent();

Title = "Главная";

CreatePage();

}

private async void CreatePage()

{

//узнаю число статей, для случайной генерации ссылок на статьи в главном меню

int num\_of\_pages = Convert.ToInt32(await client.GetStringAsync("https://apifordb20200517173817.azurewebsites.net/api/DB/GetSize"));

//это чтобы статьи не повторялись

List<int> vs = new List<int>();

//j временно нужна, пока в БД мало статей. в обновлениях уберём

int j = 0;

do

{

int number = rnd.Next(0, num\_of\_pages);

if (!vs.Contains(number))

{

vs.Add(number);

}

j++;

} while (vs.Count <= 9 && j<30);

//отсюда начинается заполнение ячеек в Grid

int row = 0;

int colomn = 0;

for (int i = 0; i < vs.Count; i++)

{

//тут ссылка для получения контента для ячейки

string json = await client.GetStringAsync($"https://apifordb20200517173817.azurewebsites.net/api/DB/GetMain/{vs[i]}");

//дессериализация полученной строки

JObject jObject = JObject.Parse(json);

MyPageInMain page = jObject.ToObject<MyPageInMain>();

//заполнение ячейки. снизу изображение, на ней черный снизу, прозрачный png файл, что также является кнопкой

Add\_Image(page.ContentsForMainImage.UrlToPict, colomn, row);

Add\_Dark(vs[i], row, colomn, page.ContentsForMainLabel.Text);

Add\_Label(page.ContentsForMainLabel.Text, colomn, row);

if (colomn == 0)

{

colomn++;

}

else

{

colomn--;

row++;

}

}

}

/// <summary>

/// добавляет изображение в ячейку

/// </summary>

/// <param name="url">ссылка на изображение</param>

/// <param name="colomn">столбец в Grid</param>

/// <param name="row">строка в Grid</param>

private void Add\_Image(string url, int colomn, int row)

{

grid.Children.Add(new Image() { VerticalOptions = LayoutOptions.End, HorizontalOptions = LayoutOptions.FillAndExpand, Aspect = Aspect.Fill, Source = new UriImageSource { Uri = new Uri($"{url}") } },

colomn, row);

}

/// <summary>

/// добавляет "затемнитель" снизу ячейки, т.к. на фоне белых изображений не видно название статьи. также является кнопкой, для перехода к статье

/// </summary>

/// <param name="id">id строки статьи в БД. передаётся на следующую страницу где используется для получения информации с сервера</param>

/// <param name="colomn">столбец в Grid</param>

/// <param name="row">строка в Grid</param>

/// <param name="title">также название статьи</param>

private void Add\_Dark(int id, int colomn, int row, string title)

{

//создание кнопки

ImageButton imageButton = new ImageButton()

{ Source = ImageSource.FromResource("IronLib.Picts.dark.png"), VerticalOptions = LayoutOptions.FillAndExpand,

HorizontalOptions = LayoutOptions.FillAndExpand, Opacity = 0.1};

//подписываю на event откртие страницы просмотра контента

imageButton.Clicked += (s, e) => { PageWithText page1 = new PageWithText(id, title); Navigation.PushAsync(page1); };

grid.Children.Add(imageButton, colomn, row);

}

/// <summary>

/// добавляет название статьи в ячейку

/// </summary>

/// <param name="text">текст в ячейке</param>

/// <param name="colomn">столбец в Grid</param>

/// <param name="row">строка в Grid</param>

private void Add\_Label(string text, int colomn, int row)

{

grid.Children.Add(new Label() { Text = text, FontAttributes = FontAttributes.Bold, FontSize = Device.GetNamedSize(NamedSize.Small, typeof(Label)), VerticalOptions = LayoutOptions.End, HorizontalOptions = LayoutOptions.Start, TextColor = Color.White, MinimumHeightRequest = 150},

colomn, row);

}

}

}

## PageWithText

## PageWithText.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"

xmlns:d="http://xamarin.com/schemas/2014/forms/design"

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

mc:Ignorable="d"

x:Class="IronLib.PageWithText">

<ContentPage.Content>

<ScrollView >

<StackLayout x:Name="stackLayout">

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

## PageWithText.xaml.cs

using ClassLib;

using Newtonsoft.Json.Linq;

using System.Collections.Generic;

using System.Net.Http;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace IronLib

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class PageWithText : ContentPage

{

static readonly HttpClient client = new HttpClient();

public PageWithText(int id, string title)

{

InitializeComponent();

Title = title;

AddAll(id, stackLayout);

}

//вынес в метод т.к.async

/// <summary>

/// метод, где запрашиваю информацию из бд

/// </summary>

/// <param name="id"></param>

/// <param name="stack"></param>

private async static void AddAll(int id, StackLayout stack)

{

//ссылка на бд

string json = await client.GetStringAsync($"https://apifordb20200517173817.azurewebsites.net/api/DB/GetContent/{id}");

//парс Json в MyPageInContent, который выведет весь контент на страницу

JArray jArray = JArray.Parse(json);

MyPageInContent page = new MyPageInContent(jArray.ToObject<List<MyContent>>());

page.AddAll(stack);

}

}

}

//также в IronLib находится папка Picts, где лежат 2 файла pgn. Dark //используется в MainPage.xaml.cs, а search для будующих обновлений

# Решение APIForDB

## Класс Startup

using APIForDB.Models.Options;

using APIForDB.Providers;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.OpenApi.Models;

namespace APIForDB

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

//Необходимо подключить конфиг, в котором содержится строка подключения к БД

services.Configure<DbConnectionOptions>(Configuration.GetSection(nameof(DbConnectionOptions)));

//Добавляем новый сервис

services.AddSingleton<DBService, DBService>();

//Подключаем провайдер подключения

services.AddSingleton<ISQLiteConnectionProvider, SQLiteConnectionProvider>();

//Добавляем репозиторий

services.AddSingleton<IDBRepository, DBRepository>();

services.AddMvc()

.SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "DBForIronLib", Version = "v1" });

});

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

//сваггер нужен поддержке

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "DBForIronLib");

c.RoutePrefix = "swagger";

});

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseMvc();

}

}

}

## Класс DBController.cs. Папка Controllers

using APIForDB.Models.DB;

using Microsoft.AspNetCore.Mvc;

using System.Collections.Generic;

namespace APIForDB.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class DBController : ControllerBase

{

private readonly DBService \_DBService;

public DBController(DBService DBService)

{

\_DBService = DBService;

}

/// <summary>

/// метод для осмотра определённой статьи по id. для пользования службой поддержки

/// </summary>

/// <returns></returns>

[HttpGet]

public GetDBResponse GetIDPageDB([FromQuery]int id)

{

var result = \_DBService.GetAllDB(id);

return new GetDBResponse

{

Pages = result.pages,

Error = result.error

};

}

/// <summary>

/// метод для осмотра всех статей. для пользования службой поддержки

/// </summary>

/// <returns></returns>

[HttpGet("GetAll")]

public GetDBResponse GetAllDB()

{

var result = \_DBService.GetAllDB();

return new GetDBResponse

{

Pages = result.pages,

Error = result.error

};

}

/// <summary>

/// добавление новых статей

/// </summary>

/// <param name="phone"></param>

/// <returns></returns>

[HttpPost]

public AddPageResponse AddPage([FromBody] AddPageRequest page)

{

return new AddPageResponse

{

Error = \_DBService.AddPage(page)?.Error

};

}

[HttpPut]

public UpdatePageResponse UpdateCustomer([FromQuery]int id, [FromBody] UpdatePageRequest request)

{

return \_DBService.UpdatePage(id, request);

}

/// <summary>

/// возвращение номеров статей по определённой тематике. Технический метод

/// </summary>

/// <param name="info"></param>

/// <returns></returns>

[HttpGet("GetByTypes/{info}")]

public IEnumerable<int> GetByTypes(string info)

{

return \_DBService.GetByTypes(info);

}

/// <summary>

/// получение размера. Технический метод для случайной генерации статей

/// </summary>

/// <returns></returns>

[HttpGet("GetSize")]

public string GetSize()

{

return \_DBService.GetSize();

}

/// <summary>

/// для получения визуальной части на главной. для пользователя

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("GetMain/{id}")]

public string GetPageDB\_User(int id)

{

return \_DBService.GetMainDB(id);

}

/// <summary>

/// для получения контента. для пользователя

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

[HttpGet("GetContent/{id}")]

public string GetContentDB\_User(int id)

{

return \_DBService.GetContentDB(id);

}

}

## Класс DBService.cs. Папка Services

using APIForDB.Models.DB;

using System.Collections.Generic;

namespace APIForDB

{

public interface IDBService

{

string GetSize();

IEnumerable<int> GetByTypes(string info);

string GetMainDB(int id);

string GetContentDB(int id);

(IEnumerable<Page> pages, string error) GetAllDB(int count = -1);

AddPageResponse AddPage(AddPageRequest page);

UpdatePageResponse UpdatePage(int id, UpdatePageRequest request);

}

public class DBService : IDBService

{

private readonly IDBRepository \_dbRepository;

public DBService(IDBRepository dbRepository)

=> \_dbRepository = dbRepository;

/// <summary>

/// транзит в репозиторий для получения информации разработчиком

/// </summary>

/// <param name="count"></param>

/// <returns></returns>

public (IEnumerable<Page> pages, string error) GetAllDB(int count = -1)

{

return \_dbRepository.GetAllDB(count);

}

/// <summary>

/// транзит в репозиторий для добавления страницы

/// </summary>

/// <param name="page"></param>

/// <returns></returns>

public AddPageResponse AddPage(AddPageRequest page)

{

return \_dbRepository.AddPage(page);

}

/// <summary>

/// транзит в репозиторий для обновления страницы

/// </summary>

/// <param name="id"></param>

/// <param name="request"></param>

/// <returns></returns>

public UpdatePageResponse UpdatePage(int id, UpdatePageRequest request)

{

return \_dbRepository.UpdatePage(id, request);

}

/// <summary>

/// транзит к получению размера

/// </summary>

/// <returns></returns>

public string GetSize()

{

return \_dbRepository.GetSize();

}

/// <summary>

/// транзит к получению массива id тех страниц, которые имеют специализированный тип страницы

/// </summary>

/// <param name="info"></param>

/// <returns></returns>

public IEnumerable<int> GetByTypes(string info)

{

return \_dbRepository.GetByTypes(info);

}

/// <summary>

/// транзит к информации о виде статьи на главной

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public string GetMainDB(int id)

{

return \_dbRepository.GetMainDB(id);

}

/// <summary>

/// транзит к контенту статьи

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public string GetContentDB(int id)

{

return \_dbRepository.GetContentDB(id);

}

}

}

## Класс DBRepository.cs. Папка Repositories

using Dapper;

using APIForDB.Providers;

using System;

using System.Collections.Generic;

using APIForDB.Models.DB;

using System.Linq;

namespace APIForDB

{

public interface IDBRepository

{

string GetSize();

IEnumerable<int> GetByTypes(string info);

string GetMainDB(int id);

string GetContentDB(int id);

(IEnumerable<Page> pages, string error) GetAllDB(int count = -1);

AddPageResponse AddPage(AddPageRequest page);

UpdatePageResponse UpdatePage(int id, UpdatePageRequest request);

}

public class DBRepository : IDBRepository

{

private readonly ISQLiteConnectionProvider \_connectionProvider;

public DBRepository(ISQLiteConnectionProvider sqliteConnectionProvider)

{

\_connectionProvider = sqliteConnectionProvider;

}

/// <summary>

/// получение всей или определённой страницы бд

/// </summary>

/// <param name="count"></param>

/// <returns></returns>

public (IEnumerable<Page> pages, string error) GetAllDB(int count = -1)

{

using (var connection = \_connectionProvider.GetDbConnection())

{

try

{

connection.Open();

if (count == -1)

{

return (connection.Query<Page>(@"

SELECT

id as Id,

page as Main,

content as Contents,

types as Type

FROM Data"), null);

}

else

{

return (

connection.Query<Page>(@"

SELECT

id as Id,

page as Main,

content as Contents,

types as Type

FROM Data

Where @count == id",

new { count = count }), null);

}

}

catch(Exception e)

{

return (null, e.Message);

}

}

}

/// <summary>

/// добавление статьи в БД. Для поддержки

/// </summary>

/// <param name="page"></param>

/// <returns></returns>

public AddPageResponse AddPage(AddPageRequest page)

{

try

{

using (var connection = \_connectionProvider.GetDbConnection())

{

connection.Open();

connection.Execute(

@"INSERT INTO Data

( page, content, types ) VALUES

( @Page, @Content, @Types );",

new { Page = page.Page, Content = page.Content, Types = page.Types });

return new AddPageResponse

{

Error = null

};

}

}

catch (Exception e)

{

return new AddPageResponse

{

Error = e.Message

};

}

}

/// <summary>

/// редакция статьи в БД. Для поддержки

/// </summary>

/// <param name="id"></param>

/// <param name="request"></param>

/// <returns></returns>

public UpdatePageResponse UpdatePage(int id, UpdatePageRequest request)

{

try

{

using (var connection = \_connectionProvider.GetDbConnection())

{

connection.Open();

if (0 == connection.Execute(

@"UPDATE Data

SET page = @Page, content = @Content, types = @Types

WHERE id = @Id;",

new { Page = request.Page, Content = request.Content, Types = request.Types, Id = id }))

{

throw new Exception("Нечего редактировать!");

}

}

return null;

}

catch (Exception e)

{

return new UpdatePageResponse

{

Error = e.Message

};

}

}

/// <summary>

/// получение числа строк

/// </summary>

/// <returns></returns>

public string GetSize()

{

using (var connection = \_connectionProvider.GetDbConnection())

{

connection.Open();

return connection.Query<string>(@"

SELECT COUNT (\*)

FROM Data").First();

}

}

/// <summary>

/// получение статей по определённому тегу. не успел реализовать в приложении. Оставил на будущее

/// </summary>

/// <param name="info"></param>

/// <returns></returns>

public IEnumerable<int> GetByTypes(string info)

{

using (var connection = \_connectionProvider.GetDbConnection())

{

connection.Open();

return connection.Query<int>(@"

SELECT

id

FROM Data

WHERE name LIKE @searchTerm",

new { searchTerm = '%' + info + '%' });

}

}

/// <summary>

/// получение json для главной по id. для пользователя

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public string GetMainDB(int id)

{

using (var connection = \_connectionProvider.GetDbConnection())

{

connection.Open();

return connection.Query<string>(@"

SELECT page

FROM Data

WHERE id = Id;", new { Id = id }).First();

}

}

/// <summary>

/// получение json для главной по id. для пользователя

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public string GetContentDB(int id)

{

using (var connection = \_connectionProvider.GetDbConnection())

{

connection.Open();

return connection.Query<string>(@"

SELECT content

FROM Data

WHERE id = Id;", new { Id = id }).First();

}

}

}

}

* 1. Класс SQLiteProvider.cs. Файл Providers

using Dapper;

using APIForDB.Models.Options;

using Microsoft.Extensions.Options;

using System.Data.SQLite;

namespace APIForDB.Providers

{

public interface ISQLiteConnectionProvider

{

SQLiteConnection GetDbConnection();

}

/// <summary>

/// тут БД создаётся и даётся к ней доступ

/// </summary>

public class SQLiteConnectionProvider : ISQLiteConnectionProvider

{

private readonly string \_connectionString;

public SQLiteConnectionProvider(IOptions<DbConnectionOptions> dbOptions)

{

\_connectionString = dbOptions.Value.ConnectionString;

using (var connection = GetDbConnection())

{

var createDB =

@"CREATE TABLE IF NOT EXISTS Data

(

id integer primary key AUTOINCREMENT,

page varchar not null,

content varchar not null,

types varchar not null

)";

connection.Execute(createDB);

}

}

public SQLiteConnection GetDbConnection()

=> new SQLiteConnection(\_connectionString);

}

}

* 1. Папка Models

## Папка DB

## Класс AddPageRequest

namespace APIForDB.Models.DB

{

/// <summary>

/// технический класс для поддержки приложения. добавление новых статей

/// </summary>

public class AddPageRequest

{

public string Page { get; set; }

public string Content { get; set; }

public string Types { get; set; }

}

}

## Класс AddPageResponse

namespace APIForDB.Models.DB

{

/// <summary>

/// класс для просмотра возможный ошибок при добавлении страницы

/// </summary>

public class AddPageResponse

{

public string Error { get; set; }

}

}

## Класс GetDBResponse

using System.Collections.Generic;

namespace APIForDB.Models.DB

{

/// <summary>

/// класс для получения всей БД. для поддержки

/// </summary>

public class GetDBResponse

{

public IEnumerable<Page> Pages { get; set; }

public string Error { get; set; }

}

}

## Класс Page

namespace APIForDB.Models.DB

{

/// <summary>

/// для удобства просмотра содержимого БД службой поддержки

/// </summary>

public class Page

{

public int Id { get; set; }

public string Main { get; set; }

public string Contents { get; set; }

public string Type { get; set; }

}

}

## Класс UpdatePageRequest

namespace APIForDB.Models.DB

{

/// <summary>

/// Обновление контента относящегося к странице. Для поддержки

/// </summary>

public class UpdatePageRequest

{

public string Page { get; set; }

public string Content { get; set; }

public string Types { get; set; }

}

}

## Класс UpdatePageResponse

namespace APIForDB.Models.DB

{

/// <summary>

/// класс для просмотра возможный ошибок при редакции страницы

/// </summary>

public class UpdatePageResponse

{

public string Error { get; set; }

}

}

## Класс DbConnectionOption. Папка Options

namespace APIForDB.Models.Options

{

/// <summary>

/// класс нужен для подключения к БД

/// </summary>

public class DbConnectionOptions

{

public string ConnectionString { get; set; }

}

}

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**   
1) ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.   
2) ГОСТ 19.301-79 Программа и методика испытаний. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.   
3) ГОСТ 19.401-78 Текст программы. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.   
4) ГОСТ 19.505-79 Руководство оператора. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.   
5) ГОСТ 19.404-79 Пояснительная записка. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.   
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.   
7) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

# ПРИЛОЖЕНИЕ 1

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |