МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор

Українського державного університету науки і технології

Анатолій РАДКЕВИЧ

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

ДЛЯ ПРЕГЛЯДУ РОЗКЛАДУ ЗАНЯТЬ УНІВЕРСИТЕТУ

Текст програми

ЛИСТ ЗАТВРЕДЖЕННЯ

1116130.01318-01 12 01-ЛЗ

Завідувач кафедри КІТ

\_\_\_\_\_\_\_\_\_\_\_ Вадим ГОРЯЧКІН

Керівник розробки

\_\_\_\_\_\_\_\_\_\_\_ Олександр ЖЕВАГО

Виконавець

\_\_\_\_\_\_\_\_\_\_\_ Владислав ЗАБОЛОТНИЙ

Нормоконтролер

\_\_\_\_\_\_\_\_\_\_\_ Світлана ВОЛКОВА

2023

ЗАТВЕРДЖЕНО

1116130.01318-01 12 01-ЛЗ

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

ДЛЯ ПРЕГЛЯДУ РОЗКЛАДУ ЗАНЯТЬ УНІВЕРСИТЕТУ

Текст програми

44165850.94002-01 12 01

Листів 6

2023

АНОТАЦІЯ

Документ 44165850.94002-01 12 01 «Мобільний додаток для перегляду розкладу занять університету. Текст програми» входить до складу програмної документації на додаток, що реалізує мобільний додаток для відображення розкладів занять університету.

У даному документі представлено текст програми написаний на мові програмування C#. Об’єм пам’яті, що займає програмнийи комплекс, складає 29,3 Мб. Конфігурація телефона стандартна. Комплекс функціоную в середовищі Android 7.0 та вище або iOS 9.0 та вище.

Програма розроблена в середовищі Visual Studio 2022 за допомогою технологій Xamarin. В якості СУБД використовується SQLite.

ЗМІСТ

[1. СХЕМА ВЗАЄМОДІЇ МОДУЛІВ 5](#_Toc136640443)

[2. ТЕКСТ ПРОГРАМИ 6](#_Toc136640444)

[2.1 Core 6](#_Toc136640445)

[2.1.1 Parser 6](#_Toc136640446)

[2.1.1.1 ClassFileParser.cs 6](#_Toc136640447)

[2.1.1.2 IFileScheduleParser.cs 8](#_Toc136640448)

[2.1.1.3 ModulFileParser.cs 8](#_Toc136640449)

[2.1.2 Services 9](#_Toc136640450)

[2.1.2.1 IScheduleProvider.cs 9](#_Toc136640451)

[2.1.2.2 ShceduleService.cs 10](#_Toc136640452)

[2.1.2.3 SiteShduleProvider 11](#_Toc136640453)

[2.1.3 FileType.cs 15](#_Toc136640454)

[2.1.4 ParseResult.cs 15](#_Toc136640455)

[2.2 Domain 15](#_Toc136640456)

[2.2.1 Enums 15](#_Toc136640457)

[2.2.1.1 ClassSubType.cs 15](#_Toc136640458)

[2.2.1.2 WeekType.cs 15](#_Toc136640459)

[2.2.2 Models 15](#_Toc136640460)

[2.2.2.1 Class.cs 15](#_Toc136640461)

[2.2.2.2 Group.cs 16](#_Toc136640462)

[2.2.2.3 Schedule.cs 16](#_Toc136640463)

[2.2.2.4 ScheduleInfo.cs 17](#_Toc136640464)

[2.2.2.5 Teacher.cs 17](#_Toc136640465)

[2.2.3 PersistenceInterface 17](#_Toc136640466)

[2.2.3.1 IDbPathProvider.cs; 17](#_Toc136640467)

[2.2.3.2 IGroupRepository.cs 18](#_Toc136640468)

[2.2.3.3 IScheduleRepository.cs 18](#_Toc136640469)

[2.2.3.4 ITeacherRepository.cs 18](#_Toc136640470)

[2.2.4 SearchCriteria.cs 18](#_Toc136640471)

[2.3 Persistence 18](#_Toc136640472)

[2.3.1 Entities 18](#_Toc136640473)

[2.3.1.1 Class.cs 19](#_Toc136640474)

[2.3.1.2 Schedule.cs 19](#_Toc136640475)

[2.3.2 SQLiteDataBase.cs 19](#_Toc136640476)

[2.3.3 SQLiteGroupRepository.cs 20](#_Toc136640477)

[2.3.4 SQLiteScheduleRepository.cs 20](#_Toc136640478)

[2.3.5 SQLiteTeacherRepoditory.cs 21](#_Toc136640479)

[2.3.6 TableName.cs 22](#_Toc136640480)

[2.4 Mobile 22](#_Toc136640481)

[2.4.1 Views 22](#_Toc136640482)

[2.4.1.1 App.xaml 22](#_Toc136640483)

[2.4.1.1.1 App.cs 22](#_Toc136640484)

[2.4.1.2 MainPage.xaml 22](#_Toc136640485)

[2.4.1.2.1 MainPage.cs 23](#_Toc136640486)

[2.4.1.3 SchedulePage.xaml 23](#_Toc136640487)

[2.4.1.3.1 SchedulePage.cs 27](#_Toc136640488)

[2.4.2 AssemblyInfo.cs 34](#_Toc136640489)

[2.4.3 DependencyInjectionContainer.cs 34](#_Toc136640490)

[2.4.4 Startup.cs; 35](#_Toc136640491)

[2.5 Mobile.Android 35](#_Toc136640492)

[2.5.1 AndroidDbPathProvider.cs 35](#_Toc136640493)

[2.5.2 MainActivity.cs 35](#_Toc136640494)

[2.6 Mobile.iOS 36](#_Toc136640495)

[2.6.1 AppDeleagte.cs 36](#_Toc136640496)

[2.6.2 iOSDbPathProvider.cs 36](#_Toc136640497)

[2.6.3 Main.cs 36](#_Toc136640498)

1. СХЕМА ВЗАЄМОДІЇ МОДУЛІВ
2. ТЕКСТ ПРОГРАМИ
   1. Core
      1. Parser
         1. ClassFileParser.cs

using Domain.Enums;

using Domain.Models;

using NPOI.HSSF.UserModel;

using NPOI.SS.UserModel;

using NPOI.XSSF.UserModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace Core.Parsers

{

public class ClassFileParser : IFileScheduleParser

{

public Task<ParseResult> Parse(byte[] fileData, FileType, string scheduleName)

{

if (fileType == FileType.NotSupported || !scheduleName.Contains("Розклад занять"))

{

return Task.FromResult(ParseResult.NotSupportedFile());

}

var result = ParseFile(fileData, fileType);

return Task.FromResult(result);

}

private ParseResult ParseFile(byte[] fileData, FileType fileType)

{

try

{

using (var ms = new MemoryStream(fileData))

{

var workbook = fileType == FileType.Xls ? new HSSFWorkbook(ms) : (IWorkbook)new XSSFWorkbook(ms);

var sheet = workbook.GetSheetAt(0);

var columnsCount = sheet.GetRow(3).Count();

var rowCount = sheet.LastRowNum;

var classes = new List<Class>();

for (int columnindex = 2; columnindex < columnsCount; columnindex++)

{

var cells = new List<ICell>();

for (var rowIndex = 3; rowIndex <= rowCount; rowIndex++)

{

var row = sheet.GetRow(rowIndex);

if (row == null)

{

rowCount = rowIndex - 1;

break;

}

var cell = row.GetCell(columnindex);

cells.Add(cell);

}

var weekDayCounter = 1;

var numberCounter = 1;

for (var cellIndex = 0; cellIndex < cells.Count; cellIndex += rowCount)

{

string groupName = string.Empty, groupAlternativeName = string.Empty;

if (cells[cellIndex].IsMergedCell)

{

var stringValue = cells[cellIndex].StringCellValue;

var groupNameParts = stringValue.Split('\n');

groupName = groupNameParts[0];

groupAlternativeName = groupNameParts[1];

}

else

{

groupName = cells[cellIndex].StringCellValue;

groupAlternativeName = cells[cellIndex + 1].StringCellValue;

}

var group = new Group(groupName, groupAlternativeName);

for (var index = cellIndex + 2; index < rowCount && index + 4 < rowCount; index += 4)

{

if (index + 5 > cells.Count)

{

break;

}

if (numberCounter > 8)

{

weekDayCounter++;

numberCounter = 1;

}

if (cells[index] == null)

{

continue;

}

if (cells[index].IsMergedCell)

{

if (!string.IsNullOrEmpty(cells[index].StringCellValue))

{

var teacher = !string.IsNullOrEmpty(cells[index + 3].StringCellValue) ? new Teacher(cells[index + 3].StringCellValue) : new Teacher("-");

var @class = new Class(cells[index].StringCellValue, group, teacher, null, weekDayCounter, numberCounter, WeekType.None);

classes.Add(@class);

}

}

else

{

if (!string.IsNullOrEmpty(cells[index].StringCellValue) )

{

var teacher = !string.IsNullOrEmpty(cells[index + 1].StringCellValue) ? new Teacher(cells[index + 1].StringCellValue) : new Teacher("-");

var @class = new Class(cells[index].StringCellValue, group, teacher, null, weekDayCounter, numberCounter, WeekType.Numerator);

classes.Add(@class);

}

if (cells[index + 2] == null)

{

continue;

}

if (!string.IsNullOrEmpty(cells[index + 2].StringCellValue))

{

var teacher = cells[index + 3].StringCellValue != "" ? new Teacher(cells[index + 3].StringCellValue) : new Teacher("-");

var @class = new Class(cells[index + 2].StringCellValue, group, teacher, null, weekDayCounter, numberCounter, WeekType.Denominator);

classes.Add(@class);

}

}

numberCounter++;

}

}

}

var res = new ParseResult(classes.ToArray(), null);

return new ParseResult(classes.ToArray(), null);

}

}

catch (Exception e)

{

return ParseResult.ParseFailed(e.Message);

}

}

}

}

* + - 1. IFileScheduleParser.cs

using System.Threading.Tasks;

namespace Core.Parsers

{

public interface IFileScheduleParser

{

Task<ParseResult> Parse(byte[] fileData, FileType, string scheduleName);

}

}

* + - 1. ModulFileParser.cs

using Domain.Enums;

using Domain.Models;

using NPOI.HSSF.UserModel;

using NPOI.SS.UserModel;

using NPOI.XSSF.UserModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

namespace Core.Parsers

{

public class ModuleFileParser : IFileScheduleParser

{

public Task<ParseResult> Parse(byte[] fileData, FileType, string scheduleName)

{

if (fileType == FileType.NotSupported || !scheduleName.Contains("МК"))

{

return Task.FromResult(ParseResult.NotSupportedFile());

}

var result = ParseFile(fileData, fileType);

return Task.FromResult(result);

}

private ParseResult ParseFile(byte[] fileData, FileType fileType)

{

try

{

using (var ms = new MemoryStream(fileData))

{

var workbook = fileType == FileType.Xls ? new HSSFWorkbook(ms) : (IWorkbook)new XSSFWorkbook(ms);

int countSheet = 2;

var classes = new List<Class>();

for (var sheetIndex = 0; sheetIndex < countSheet; sheetIndex++)

{

var sheet = workbook.GetSheetAt(sheetIndex);

var columnsCount = sheet.GetRow(3).Count();

var rowCount = sheet.LastRowNum;

for (var columnIndex = 0; columnIndex < columnsCount; columnIndex += 2)

{

var cells = new List<ICell>();

for (var rowIndex = 3; rowIndex <= rowCount; rowIndex++)

{

var row = sheet.GetRow(rowIndex);

if (row == null)

{

rowCount = rowIndex - 1;

break;

}

var cell = row.GetCell(columnIndex);

cells.Add(cell);

}

for (var cellIndex = 0; cellIndex < cells.Count; cellIndex += rowCount)

{

var stringValue = cells[cellIndex].StringCellValue;

var groupNameParts = stringValue.Split('\n');

var group = new Group(groupNameParts[0], groupNameParts[1]);

for (var index = cellIndex + 1; index < rowCount && index + 5 < rowCount; index += 5)

{

if (index + 6 > cells.Count)

{

break;

}

if (cells[index] != null

&& cells[index + 1] != null

&& cells[index + 2] != null

&& cells[index + 3] != null

&& cells[index + 4].StringCellValue != ""

&& !string.IsNullOrEmpty(cells[index + 2].StringCellValue))

{

var subject = cells[index + 2].StringCellValue;

var teacher = new Teacher(cells[index + 3].StringCellValue);

var date = cells[index].DateCellValue;

var timeStringValue = cells[index + 4].StringCellValue;

var timeParts = timeStringValue.Split('-');

var fullDate = new DateTime(date.Year, date.Month, date.Day, int.Parse(timeParts[0]), int.Parse(timeParts[1]), 0);

var number = GetClassNumber(timeStringValue);

var subType = cells[index + 1].StringCellValue == "Консультація" ? ClassSubType.Consultation : ClassSubType.Exam;

var @class = new Class(subject, group, teacher, null, (int)date.DayOfWeek, number, WeekType.None, subType, fullDate);

classes.Add(@class);

}

}

}

}

}

var res = new ParseResult(classes.ToArray(), null);

return new ParseResult(classes.ToArray(), null);

}

}

catch (Exception e)

{

return ParseResult.ParseFailed(e.Message);

}

}

public static int GetClassNumber(string str)

{

switch (str)

{

case "8-00":

return 1;

case "9-30":

return 2;

case "11-00":

return 3;

case "13-00":

return 4;

case "14-30":

return 5;

case "16-00":

return 6;

default:

return 0;

}

}

}

}

* + 1. Services
       1. IScheduleProvider.cs

using Domain.Models;

using System.Threading.Tasks;

namespace Core.Services

{

public interface IScheduleProvider

{

Task<ScheduleInfo[]> GetScheduleInfos();

Task<ScheduleInfo> ExtendScheduleInfo(ScheduleInfo initialScheduleInfo);

Task<Schedule> Get(ScheduleInfo scheduleInfo);

}

}

* + - 1. ShceduleService.cs

using Domain;

using Domain.Models;

using Domain.PersistenceInterfaces;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Core.Services

{

public class ScheduleService

{

private readonly IScheduleProvider scheduleProvider;

private readonly IScheduleRepository scheduleRepository;

private readonly ITeacherRepository teacherRepository;

private readonly IGroupRepository groupRepository;

#region Cache

private static readonly ConcurrentDictionary<string, string[]> teacherNamesCache = new ConcurrentDictionary<string, string[]>();

private static readonly ConcurrentDictionary<string, string[]> groupNamesCache = new ConcurrentDictionary<string, string[]>();

#endregion Cache

public ScheduleService(

IScheduleProvider scheduleProvider,

IScheduleRepository scheduleRepository,

ITeacherRepository teacherRepository,

IGroupRepository groupRepository)

{

this.scheduleProvider = scheduleProvider;

this.scheduleRepository = scheduleRepository;

this.teacherRepository = teacherRepository;

this.groupRepository = groupRepository;

}

public async Task<ScheduleInfo[]> GetScheduleInfos()

{

var scheduleInfos = await scheduleProvider.GetScheduleInfos();

if (scheduleInfos?.Any() != true)

{

scheduleInfos = await scheduleRepository.Get();

}

return scheduleInfos.OrderByDescending(x => x.Year).ThenByDescending(x => x.Name).ToArray();

}

public async Task LoadSchedule(ScheduleInfo scheduleInfo)

{

var extendedScheduleInfo = await scheduleProvider.ExtendScheduleInfo(scheduleInfo) ?? scheduleInfo;

var dbScheduleInfo = await scheduleRepository.GetInfo(scheduleInfo.Name);

if (dbScheduleInfo == null)

{

var schedule = await scheduleProvider.Get(extendedScheduleInfo);

await scheduleRepository.Create(schedule);

}

else if (!string.Equals(dbScheduleInfo.Checksum, extendedScheduleInfo.Checksum))

{

var schedule = await scheduleProvider.Get(extendedScheduleInfo);

await scheduleRepository.Delete(extendedScheduleInfo.Name);

await scheduleRepository.Create(schedule);

}

}

public Task<IEnumerable<Class>> GetClasses(SearchCriteria searchCriteria) => scheduleRepository.GetClasses(searchCriteria);

public async Task<string[]> GetTeachers(string scheduleName)

{

if (teacherNamesCache.TryGetValue(scheduleName, out var teacherNames))

{

return teacherNames;

}

var teachers = await teacherRepository.Get(scheduleName);

var sortedResult = teachers.OrderBy(x => x.Name).Select(t => t.Name).ToArray();

teacherNamesCache.TryAdd(scheduleName, sortedResult);

return sortedResult;

}

public async Task<string[]> GetGroups(string scheduleName)

{

if (groupNamesCache.TryGetValue(scheduleName, out var groupNames))

{

return groupNames;

}

var groups = await groupRepository.Get(scheduleName);

var sortedResult = groups.OrderBy(x => x.Name).Select(t => t.Name).ToArray();

groupNamesCache.TryAdd(scheduleName, sortedResult);

return sortedResult;

}

}

}

* + - 1. SiteShduleProvider

using Core.Parsers;

using Domain.Models;

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Security.Cryptography;

using System.Threading.Tasks;

namespace Core.Services

{

public class SiteScheduleProvider : IScheduleProvider

{

private readonly IEnumerable<IFileScheduleParser> fileScheduleParsers;

private static readonly Uri siteUri = new Uri("https://ust.edu.ua/student/lessons\_schedule");

#region Cache

private static readonly ConcurrentDictionary<Uri, byte[]> fileCache = new ConcurrentDictionary<Uri, byte[]>();

private static ScheduleInfo[] siteSchedulesCache;

private static readonly ConcurrentDictionary<string, ScheduleInfo> scheduleInfoCache = new ConcurrentDictionary<string, ScheduleInfo>();

private static readonly ConcurrentDictionary<string, Schedule> scheduleCache = new ConcurrentDictionary<string, Schedule>();

#endregion Cache

public SiteScheduleProvider(IEnumerable<IFileScheduleParser> fileScheduleParsers)

{

this.fileScheduleParsers = fileScheduleParsers;

}

public async Task<ScheduleInfo[]> GetScheduleInfos()

{

if (siteSchedulesCache != null)

{

return siteSchedulesCache;

}

var result = await GetInfos();

if (result.Any())

{

siteSchedulesCache = result;

}

return result;

}

public async Task<ScheduleInfo> ExtendScheduleInfo(ScheduleInfo initialScheduleInfo)

{

if (scheduleInfoCache.TryGetValue(initialScheduleInfo.Name, out var scheduleInfo))

{

return scheduleInfo;

}

if (initialScheduleInfo.Url == null)

{

return initialScheduleInfo;

}

var fileData = await GetFileData(initialScheduleInfo.Url);

if (fileData == null)

{

return initialScheduleInfo;

}

var checksum = GetCheckSum(fileData);

var result = new ScheduleInfo(initialScheduleInfo.Name, initialScheduleInfo.Year, initialScheduleInfo.Url, checksum);

scheduleInfoCache.TryAdd(initialScheduleInfo.Name, result);

return result;

}

public async Task<Schedule> Get(ScheduleInfo scheduleInfo)

{

if (scheduleCache.TryGetValue(scheduleInfo.Name, out var scheduleFromCache))

{

return scheduleFromCache;

}

var result = new Schedule(scheduleInfo, null);

if (string.IsNullOrEmpty(scheduleInfo.Checksum) || scheduleInfo.Url == null)

{

return result;

}

var fileData = await GetFileData(scheduleInfo.Url);

if (fileData?.Any() != true)

{

return result;

}

var fileType = GetFileType(scheduleInfo.Url.ToString());

foreach (var fileScheduleParser in fileScheduleParsers)

{

var parseResult = await fileScheduleParser.Parse(fileData, fileType, scheduleInfo.Name);

if (!string.IsNullOrEmpty(parseResult.ErrorMessage))

{

return result;

}

if (!parseResult.NotSupported)

{

var scheduleResult = new Schedule(scheduleInfo, parseResult.Classes);

scheduleCache.TryAdd(scheduleInfo.Name, scheduleResult);

return scheduleResult;

}

}

return result;

}

private static FileType GetFileType(string url)

{

if (url.EndsWith(".xls"))

{

return FileType.Xls;

}

if (url.EndsWith(".xlsx"))

{

return FileType.Xlsx;

}

return FileType.NotSupported;

}

private string GetYear(string name)

{

var nameParts = name.Split('/');

if (nameParts.Length == 2)

{

var yearFirstPart = nameParts[0].Split(' ').LastOrDefault();

var yearSecondPart = nameParts[1].Split(' ').FirstOrDefault();

if (yearFirstPart != null && yearSecondPart != null)

{

return $"{yearFirstPart}/{yearSecondPart}";

}

}

return null;

}

private async Task<byte[]> GetFileData(Uri uri)

{

if (fileCache.TryGetValue(uri, out var data))

{

return data;

}

try

{

using (var client = new HttpClient())

{

var response = await client.GetAsync(uri);

if (response.IsSuccessStatusCode)

{

var result = await response.Content.ReadAsByteArrayAsync();

fileCache.TryAdd(uri, result);

return result;

}

}

}

catch (Exception)

{

}

return null;

}

private string GetCheckSum(byte[] data)

{

var md5 = MD5.Create();

var hash = md5.ComputeHash(data);

return BitConverter.ToString(hash).Replace("-", "");

}

private async Task<ScheduleInfo[]> GetInfos()

{

try

{

var handler = new HttpClientHandler

{

AllowAutoRedirect = false,

AutomaticDecompression = System.Net.DecompressionMethods.Deflate | System.Net.DecompressionMethods.GZip | System.Net.DecompressionMethods.None

};

using (var client = new HttpClient(handler))

{

var response = await client.GetAsync(siteUri);

if (!response.IsSuccessStatusCode)

{

return Array.Empty<ScheduleInfo>();

}

var html = await response.Content.ReadAsStringAsync();

if (string.IsNullOrEmpty(html))

{

return Array.Empty<ScheduleInfo>();

}

var parseResult = ParseSiteHtml(html);

var infos = new List<ScheduleInfo>();

foreach (var item in parseResult)

{

var info = new ScheduleInfo(item.Key, GetYear(item.Key), new Uri(item.Value), null);

infos.Add(info);

}

return infos.ToArray();

}

}

catch (Exception)

{

return Array.Empty<ScheduleInfo>();

}

}

private IDictionary<string, string> ParseSiteHtml(string html)

{

var result = new Dictionary<string, string>();

var doc = new HtmlAgilityPack.HtmlDocument();

doc.LoadHtml(html);

var nodes = doc.DocumentNode.SelectNodes(".//div[@class='col-xs-12']//div[@class='content']");

if (nodes?.Any() != true)

{

return result;

}

foreach (var node in nodes)

{

var linkNodes = node.SelectNodes(".//a");

if (linkNodes?.Any() == true)

{

foreach (var linkNode in linkNodes)

{

var href = linkNode.GetAttributeValue("href", "");

if (!string.IsNullOrEmpty(href))

{

var scheduleName = linkNode.InnerText.Replace("&nbsp;", " ");

string scheduleUrl;

if (!href.StartsWith("http"))

{

if (href.Contains("////"))

scheduleUrl = href.Replace("////", "https://");

else

scheduleUrl = "https:" + href;

}

else

{

scheduleUrl = href;

}

if (scheduleUrl.EndsWith(".xls") || scheduleUrl.EndsWith(".xlsx"))

{

result.Add(scheduleName, scheduleUrl);

}

}

}

}

}

return result;

}

}

}

* + 1. FileType.cs

namespace Core

{

public enum FileType

{

Xls,

Xlsx,

NotSupported

}

}

* + 1. ParseResult.cs

using Domain.Models;

using System;

using System.Linq;

namespace Core

{

public class ParseResult

{

public Class[] Classes { get; }

public string ErrorMessage { get; }

public bool NotSupported => (Classes?.Any() != true) && string.IsNullOrEmpty(ErrorMessage);

public ParseResult(Class[] classes, string errorMessage)

{

Classes = classes;

ErrorMessage = errorMessage;

}

public static ParseResult NotSupportedFile()

{

return new ParseResult(Array.Empty<Class>(), null);

}

public static ParseResult ParseFailed(string errorMessage)

{

return new ParseResult(Array.Empty<Class>(), errorMessage);

}

}

}

* 1. Domain
     1. Enums
        1. ClassSubType.cs

namespace Domain.Enums

{

public enum ClassSubType

{

Exam,

Consultation,

Other

}

}

* + - 1. WeekType.cs

namespace Domain.Enums

{

public enum WeekType

{

None,

Numerator,

Denominator

}

}

* + 1. Models
       1. Class.cs

using Domain.Enums;

using System;

namespace Domain.Models

{

public class Class

{

public string Subject { get; }

public Group Group { get; }

public Teacher Teacher { get; }

public string Auditory { get; }

public int WeekDay { get; }

public int Number { get; }

public WeekType WeekType { get; }

public ClassSubType SubType { get; }

public DateTime? Date { get; }

public Class(string subject, Group group, Teacher teacher, string auditory, int weekDay, int number, WeekType weekType, ClassSubType subType, DateTime? date)

{

if (string.IsNullOrEmpty(subject))

{

throw new ArgumentNullException(nameof(subject));

}

if (subType != ClassSubType.Other && date == null)

{

throw new ArgumentNullException(nameof(date));

}

Group = group ?? throw new ArgumentNullException(nameof(group));

WeekDay = weekDay;

Number = number;

WeekType = weekType;

Subject = subject;

Teacher = teacher;

Auditory = auditory;

SubType = subType;

Date = date;

}

public Class(string subject, Group group, Teacher teacher, string auditory, int weekDay, int number, WeekType weekType)

: this(subject, group, teacher, auditory, weekDay, number, weekType, ClassSubType.Other, null)

{ }

public override string ToString()

{

return $"{Subject} {Group.Name}";

}

}

}

* + - 1. Group.cs

using System;

using System.Collections.Generic;

namespace Domain.Models

{

public class Group

{

public string Name { get; }

public string AlternativeName { get; }

public Group(string name, string alternativeName)

{

if (string.IsNullOrEmpty(name))

{

throw new ArgumentNullException(nameof(name));

}

Name = name;

AlternativeName = alternativeName;

}

public override bool Equals(object obj) => obj is Group group && Name == group.Name;

public override int GetHashCode() => 539060726 + EqualityComparer<string>.Default.GetHashCode(Name);

}

}

* + - 1. Schedule.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Domain.Models

{

public class Schedule

{

public ScheduleInfo Info { get; }

public IEnumerable<Class> Classes { get; }

public Schedule(ScheduleInfo info, IEnumerable<Class> classes)

{

if (info == null)

{

throw new ArgumentNullException(nameof(info));

}

Classes = classes?.ToList() ?? new List<Class>();

Info = info;

}

}

}

* + - 1. ScheduleInfo.cs

using System;

namespace Domain.Models

{

public class ScheduleInfo

{

public string Name { get; }

public string Year { get; }

public Uri Url { get; }

public string Checksum { get; }

public ScheduleInfo(string name, string year, Uri url, string checksum)

{

if (string.IsNullOrEmpty(name))

{

throw new ArgumentNullException(nameof(name));

}

Name = name;

Year = year;

Url = url;

Checksum = checksum;

}

}

}

* + - 1. Teacher.cs

using System;

using System.Collections.Generic;

namespace Domain.Models

{

public class Teacher

{

public string Name { get; }

public Teacher(string name)

{

if (string.IsNullOrEmpty(name))

{

throw new ArgumentNullException(nameof(name));

}

Name = name;

}

public override bool Equals(object obj) => obj is Teacher teacher && Name == teacher.Name;

public override int GetHashCode() => 539060726 + EqualityComparer<string>.Default.GetHashCode(Name);

}

}

* + 1. PersistenceInterface
       1. IDbPathProvider.cs;

namespace Domain.PersistenceInterfaces

{

public interface IDbPathProvider

{

string Path { get; }

}

}

* + - 1. IGroupRepository.cs

using Domain.Models;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace Domain.PersistenceInterfaces

{

public interface IGroupRepository

{

Task<IEnumerable<Group>> Get(string scheduleName);

}

}

* + - 1. IScheduleRepository.cs

using Domain.Models;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace Domain.PersistenceInterfaces

{

public interface IScheduleRepository

{

Task<ScheduleInfo[]> Get();

Task<ScheduleInfo> GetInfo(string name);

Task<IEnumerable<Class>> GetClasses(string name);

Task<IEnumerable<Class>> GetClasses(SearchCriteria searchCriteria);

Task Delete(string name);

Task Create(Schedule schedule);

}

}

* + - 1. ITeacherRepository.cs

using Domain.Models;

using System.Collections.Generic;

using System.Threading.Tasks;

namespace Domain.PersistenceInterfaces

{

public interface ITeacherRepository

{

Task<IEnumerable<Teacher>> Get(string scheduleName);

}

}

* + 1. SearchCriteria.cs

using System;

namespace Domain

{

public class SearchCriteria

{

public string ScheduleName { get; }

public string TeacherName { get; }

public string GroupName { get; }

public SearchCriteria(string scheduleName, string teacherName, string groupName)

{

if (string.IsNullOrEmpty(scheduleName))

{

throw new ArgumentNullException(nameof(scheduleName));

}

ScheduleName = scheduleName;

TeacherName = teacherName;

GroupName = groupName;

}

}

}

* 1. Persistence
     1. Entities
        1. Class.cs

using SQLite;

namespace Persistence.SQLiteDb.Entities

{

[Table(TableNames.ClassTableName)]

public class Class

{

[NotNull]

public string ScheduleName { get; set; }

[NotNull]

public string Subject { get; set; }

[NotNull]

public string GroupName { get; set; }

public string GroupAlternativeName { get; set; }

public string TeacherName { get; set; }

public string Auditory { get; set; }

public int WeekDay { get; set; }

public int Number { get; set; }

[NotNull]

public string WeekType { get; set; }

[NotNull]

public string SubType { get; set; }

public long? Timestamp { get; set; }

}

}

* + - 1. Schedule.cs

using SQLite;

namespace Persistence.SQLiteDb.Entities

{

[Table(TableNames.ScheduleTableName)]

public class Schedule

{

[PrimaryKey]

public string Name { get; set; }

public string Year { get; set; }

public string Url { get; set; }

public string Checksum { get; set; }

}

}

* + 1. SQLiteDataBase.cs

using Domain.PersistenceInterfaces;

using Persistence.SQLiteDb.Entities;

using SQLite;

namespace Persistence.SQLiteDb

{

public class SQLiteDatabase

{

public SQLiteAsyncConnection Connection { get; }

public SQLiteDatabase(IDbPathProvider dbPathProvider)

{

Connection = new SQLiteAsyncConnection(dbPathProvider.Path, SQLiteOpenFlags.Create | SQLiteOpenFlags.ReadWrite | SQLiteOpenFlags.FullMutex);

InitializeDatabase();

}

private void InitializeDatabase()

{

Connection.CreateTableAsync<Class>().Wait();

Connection.CreateTableAsync<Schedule>().Wait();

}

}

}

* + 1. SQLiteGroupRepository.cs

using Domain.Models;

using Domain.PersistenceInterfaces;

using SQLite;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Persistence.SQLiteDb

{

public class SQLiteGroupRepository : IGroupRepository

{

private readonly SQLiteAsyncConnection database;

public SQLiteGroupRepository(SQLiteDatabase sQLiteDatabase)

{

database = sQLiteDatabase.Connection;

}

public async Task<IEnumerable<Group>> Get(string scheduleName)

{

var groups = await database.QueryAsync<GroupData>($"SELECT DISTINCT {nameof(GroupData.GroupName)} FROM {TableNames.ClassTableName} WHERE {nameof(Entities.Class.ScheduleName)} = ?", scheduleName);

return groups.Select(g => new Group(g.GroupName, null)).ToArray();

}

}

public class GroupData

{

public string GroupName { get; set; }

}

}

* + 1. SQLiteScheduleRepository.cs

using Domain;

using Domain.Enums;

using Domain.Models;

using Domain.PersistenceInterfaces;

using SQLite;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Persistence.SQLiteDb

{

public class SQLiteScheduleRepository : IScheduleRepository

{

private readonly SQLiteAsyncConnection database;

public SQLiteScheduleRepository(SQLiteDatabase sQLiteDatabase)

{

database = sQLiteDatabase.Connection;

}

public async Task Create(Schedule schedule)

{

var record = new Entities.Schedule

{

Checksum = schedule.Info.Checksum,

Name = schedule.Info.Name,

Url = schedule.Info.Url?.ToString(),

Year = schedule.Info.Year

};

await database.InsertAsync(record);

var classRecords = schedule.Classes.Select(c => new Entities.Class

{

Auditory = c.Auditory,

GroupAlternativeName = c.Group.AlternativeName,

GroupName = c.Group.Name,

Number = c.Number,

ScheduleName = schedule.Info.Name,

Subject = c.Subject,

SubType = c.SubType.ToString(),

TeacherName = c.Teacher?.Name,

WeekDay = c.WeekDay,

WeekType = c.WeekType.ToString(),

Timestamp = c.Date?.Ticks

}).ToList();

await database.InsertAllAsync(classRecords);

}

public async Task Delete(string name)

{

await database.Table<Entities.Class>().Where(c => c.ScheduleName == name).DeleteAsync();

await database.DeleteAsync<Entities.Schedule>(name);

}

public async Task<ScheduleInfo[]> Get()

{

var records = await database.Table<Entities.Schedule>().ToListAsync();

return records.Select(Translate).ToArray();

}

public async Task<IEnumerable<Class>> GetClasses(string name)

{

var records = await database.Table<Entities.Class>().Where(c => c.ScheduleName == name).ToListAsync();

return records.Select(Translate).ToArray();

}

public async Task<IEnumerable<Class>> GetClasses(SearchCriteria searchCriteria)

{

var query = database.Table<Entities.Class>().Where(c => c.ScheduleName == searchCriteria.ScheduleName);

if (!string.IsNullOrEmpty(searchCriteria.TeacherName))

{

query = query.Where(c => c.TeacherName == searchCriteria.TeacherName);

}

if (!string.IsNullOrEmpty(searchCriteria.GroupName))

{

query = query.Where(c => c.GroupName == searchCriteria.GroupName);

}

var records = await query.ToListAsync();

return records.Select(Translate).ToArray();

}

public async Task<ScheduleInfo> GetInfo(string name)

{

var record = await database.Table<Entities.Schedule>().FirstOrDefaultAsync(s => s.Name == name);

return record == null ? null : Translate(record);

}

private Class Translate(Entities.Class @class)

{

return new Class(

@class.Subject,

new Group(@class.GroupName, @class.GroupAlternativeName),

new Teacher(@class.TeacherName),

@class.Auditory,

@class.WeekDay,

@class.Number,

(WeekType)Enum.Parse(typeof(WeekType), @class.WeekType),

(ClassSubType)Enum.Parse(typeof(ClassSubType), @class.SubType),

@class.Timestamp == null ? null : new DateTime(@class.Timestamp.Value));

}

private ScheduleInfo Translate(Entities.Schedule schedule)

{

return new ScheduleInfo(schedule.Name, schedule.Year, string.IsNullOrEmpty(schedule.Url) ? null : new Uri(schedule.Url), schedule.Checksum);

}

}

}

* + 1. SQLiteTeacherRepoditory.cs

using Domain.Models;

using Domain.PersistenceInterfaces;

using SQLite;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Persistence.SQLiteDb

{

public class SQLiteTeacherRepository : ITeacherRepository

{

private readonly SQLiteAsyncConnection database;

public SQLiteTeacherRepository(SQLiteDatabase sQLiteDatabase)

{

database = sQLiteDatabase.Connection;

}

public async Task<IEnumerable<Teacher>> Get(string scheduleName)

{

var teachers = await database.QueryAsync<TeacherData>($"SELECT DISTINCT {nameof(TeacherData.TeacherName)} FROM {TableNames.ClassTableName} WHERE {nameof(Entities.Class.ScheduleName)} = ?", scheduleName);

return teachers.Select(g => new Teacher(g.TeacherName)).ToArray();

}

}

public class TeacherData

{

public string TeacherName { get; set; }

}

}

* + 1. TableName.cs

namespace Persistence.SQLiteDb

{

public static class TableNames

{

public const string ClassTableName = "Classes";

public const string ScheduleTableName = "Schedules";

}

}

* 1. Mobile
     1. Views
        1. 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="Mobile.App">

<Application.Resources>

</Application.Resources>

</Application>

* + - * 1. App.cs

using Microsoft.Extensions.DependencyInjection;

using System;

using Xamarin.Forms;

namespace Mobile

{

public partial class App : Application

{

public App(Action<IServiceCollection> addPlatformSpecificServices)

{

InitializeComponent();

Startup.Init(addPlatformSpecificServices);

MainPage = new NavigationPage(new MainPage());

}

protected override void OnStart()

{

}

protected override void OnSleep()

{

}

protected override void OnResume()

{

}

}

}

* + - 1. 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"

x:Class="Mobile.MainPage"

Title="УДУНТ">

<StackLayout Padding="10,10,10,10">

<Frame BackgroundColor="#2196F3" Padding="24" CornerRadius="0">

<Label Text="Оберіть розклад" HorizontalTextAlignment="Center" TextColor="White" FontSize="32" />

</Frame>

<ListView x:Name="listView" ItemSelected="onScheduleSelected" HorizontalOptions="Center">

</ListView>

</StackLayout>

</ContentPage>

* + - * 1. MainPage.cs

using Core.Services;

using Domain.Models;

using Microsoft.Extensions.DependencyInjection;

using NPOI.HSSF.Record;

using System.Linq;

using Xamarin.Forms;

namespace Mobile

{

public partial class MainPage : ContentPage

{

private readonly ScheduleService scheduleService;

private ScheduleInfo[] scheduleInfos;

public MainPage()

{

InitializeComponent();

scheduleService = Startup.ServiceProvider.GetService<ScheduleService>();

}

protected override async void OnAppearing()

{

scheduleInfos = await scheduleService.GetScheduleInfos();

listView.ItemsSource = scheduleInfos.Select(s => s.Name);

listView.SelectedItem = "";

}

private async void onScheduleSelected(object sender, SelectedItemChangedEventArgs e)

{

var selectedScheduleName = e.SelectedItem.ToString();

if (selectedScheduleName != "")

{

if (selectedScheduleName.Contains("Розклад занять") || selectedScheduleName.Contains("МК"))

{

var selectIndex = e.SelectedItemIndex;

await scheduleService.LoadSchedule(scheduleInfos[selectIndex]);

await Navigation.PushAsync(new SchedulePage(selectedScheduleName));

}

else

{

await DisplayAlert("Увага!", "Відображення даного розкладу знаходиться в розробці!", "OK");

}

}

}

}

}

* + - 1. SchedulePage.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="Mobile.SchedulePage"

Title="Розклад занять"

x:Name="schedulePage">

<StackLayout Padding="10,10,10,10">

<ScrollView>

<StackLayout>

<StackLayout>

<StackLayout Orientation="Horizontal" HorizontalOptions="Center">

<RadioButton x:Name="groupRadioButton" Content ="Студент" IsChecked="True" CheckedChanged="onScheduleTypeChange" />

<RadioButton x:Name="teacherRadioButton" Content ="Викладач" IsChecked="False" CheckedChanged="onScheduleTypeChange" />

</StackLayout>

<StackLayout>

<Picker x:Name="groupPicker" SelectedIndexChanged="onSelectedGroupChange" Title="Оберіть групу" >

<Picker.Items>

</Picker.Items>

</Picker>

<Picker x:Name="teacherPicker" SelectedIndexChanged="onSelectedTeacherChange" Title="Оберіть викладача" IsVisible="False">

<Picker.Items>

</Picker.Items>

</Picker>

</StackLayout>

</StackLayout>

<StackLayout Orientation="Horizontal">

<CheckBox x:Name="mondayCheckbox" CheckedChanged="onMondayCheckChange" />

<Label Text="Понеділок" FontSize="26" FontAttributes="Bold" />

</StackLayout>

<Frame x:Name="mondayFrame" IsVisible="false" BackgroundColor="Gray">

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="I" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="monday\_1Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="II" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="monday\_2Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="III" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="monday\_3Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="IV" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="monday\_4Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="V" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="monday\_5Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="VI" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="monday\_6Class\_StackLayout">

</StackLayout>

</StackLayout>

</StackLayout>

</Frame>

<StackLayout Orientation="Horizontal">

<CheckBox x:Name="tuesdayCheckbox" CheckedChanged="onTuesdayCheckChange" />

<Label Text="Вівторок" FontSize="26" FontAttributes="Bold" />

</StackLayout>

<Frame x:Name="tuesdayFrame" IsVisible="false" BackgroundColor="Gray">

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="I" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="tuesday\_1Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="II" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="tuesday\_2Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="III" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="tuesday\_3Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="IV" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="tuesday\_4Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="V" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="tuesday\_5Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="VI" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="tuesday\_6Class\_StackLayout">

</StackLayout>

</StackLayout>

</StackLayout>

</Frame>

<StackLayout Orientation="Horizontal">

<CheckBox x:Name="wednesdayCheckbox" CheckedChanged="onWednesdayCheckChange" />

<Label Text="Середа" FontSize="26" FontAttributes="Bold" />

</StackLayout>

<Frame x:Name="wednesdayFrame" IsVisible="false" BackgroundColor="Gray">

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="I" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="wednesday\_1Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="II" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="wednesday\_2Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="III" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="wednesday\_3Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="IV" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="wednesday\_4Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="V" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="wednesday\_5Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="VI" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="wednesday\_6Class\_StackLayout">

</StackLayout>

</StackLayout>

</StackLayout>

</Frame>

<StackLayout Orientation="Horizontal">

<CheckBox x:Name="thursdayCheckbox" CheckedChanged="onThursdayCheckChange" />

<Label Text="Четвер" FontSize="26" FontAttributes="Bold" />

</StackLayout>

<Frame x:Name="thursdayFrame" IsVisible="false" BackgroundColor="Gray">

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="I" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="thursday\_1Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="II" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="thursday\_2Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="III" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="thursday\_3Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="IV" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="thursday\_4Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="V" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="thursday\_5Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="VI" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="thursday\_6Class\_StackLayout">

</StackLayout>

</StackLayout>

</StackLayout>

</Frame>

<StackLayout Orientation="Horizontal">

<CheckBox x:Name="fridayCheckbox" CheckedChanged="onFridayCheckChange" />

<Label Text="П'ятниця" FontSize="26" FontAttributes="Bold" />

</StackLayout>

<Frame x:Name="fridayFrame" IsVisible="false" BackgroundColor="Gray">

<StackLayout>

<StackLayout Orientation="Horizontal">

<Label Text="I" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="friday\_1Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="II" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="friday\_2Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="III" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="friday\_3Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="IV" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="friday\_4Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="V" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="friday\_5Class\_StackLayout">

</StackLayout>

</StackLayout>

<BoxView Color="#FCFCFC" HeightRequest="2" HorizontalOptions="Fill" />

<StackLayout Orientation="Horizontal">

<Label Text="VI" FontSize="22" FontAttributes="Bold" WidthRequest="25" />

<BoxView Color="#FCFCFC" WidthRequest="2" HorizontalOptions="Fill" />

<StackLayout x:Name="friday\_6Class\_StackLayout">

</StackLayout>

</StackLayout>

</StackLayout>

</Frame>

</StackLayout>

</ScrollView>

</StackLayout>

</ContentPage>

* + - * 1. SchedulePage.cs

using Core.Services;

using Domain;

using Domain.Enums;

using Domain.Models;

using System;

using System.Linq;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

namespace Mobile

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class SchedulePage : ContentPage

{

private readonly ScheduleService scheduleService;

private const int maxWeekday = 5;

private const int maxNumber = 6;

private StackLayout[,] sls;

public readonly string scheduleName;

public SchedulePage(string scheduleName)

{

InitializeComponent();

this.scheduleName = scheduleName;

scheduleService = Startup.ServiceProvider.GetService<ScheduleService>();

sls = new StackLayout[,]

{

{monday\_1Class\_StackLayout, monday\_2Class\_StackLayout, monday\_3Class\_StackLayout, monday\_4Class\_StackLayout, monday\_5Class\_StackLayout, monday\_6Class\_StackLayout},

{tuesday\_1Class\_StackLayout, tuesday\_2Class\_StackLayout, tuesday\_3Class\_StackLayout, tuesday\_4Class\_StackLayout, tuesday\_5Class\_StackLayout, tuesday\_6Class\_StackLayout},

{wednesday\_1Class\_StackLayout, wednesday\_2Class\_StackLayout, wednesday\_3Class\_StackLayout, wednesday\_4Class\_StackLayout, wednesday\_5Class\_StackLayout, wednesday\_6Class\_StackLayout},

{thursday\_1Class\_StackLayout, thursday\_2Class\_StackLayout, thursday\_3Class\_StackLayout, thursday\_4Class\_StackLayout, thursday\_5Class\_StackLayout, thursday\_6Class\_StackLayout},

{friday\_1Class\_StackLayout, friday\_2Class\_StackLayout, friday\_3Class\_StackLayout, friday\_4Class\_StackLayout, friday\_5Class\_StackLayout, friday\_6Class\_StackLayout}

};

}

protected override async void OnAppearing()

{

schedulePage.Title = scheduleName;

await InitializePickers();

}

private async Task InitializePickers()

{

var groups = await scheduleService.GetGroups(scheduleName);

foreach (var groupName in groups)

{

groupPicker.Items.Add(groupName);

}

var teachers = await scheduleService.GetTeachers(scheduleName);

foreach (var teacherName in teachers)

{

if (teacherName == "-")

continue;

teacherPicker.Items.Add(teacherName);

}

}

private async Task ViewScheduleGroup(string group)

{

CleanSL(sls);

var searchCriteria = new SearchCriteria(scheduleName, null, group);

var groupClasses = await scheduleService.GetClasses(searchCriteria);

foreach (var groupClass in groupClasses)

{

for (int weekDayCounter = 1; weekDayCounter <= maxWeekday; weekDayCounter++)

{

if (groupClass.WeekDay == weekDayCounter)

{

for (int numberCounter = 1; numberCounter <= maxNumber; numberCounter++)

{

if (groupClass.Number == numberCounter)

{

ViewSubjectsGroup(sls, groupClass, weekDayCounter, numberCounter);

}

}

}

}

}

}

public void ViewSubjectsGroup(StackLayout[,] sls, Class groupClass, int weekDayCounter, int numberCounter)

{

Label labelSeparator = new Label();

labelSeparator.Text = "----------------------------------------------";

labelSeparator.FontAttributes = FontAttributes.Bold;

StackLayout sl;

BoxView box = new BoxView

{

WidthRequest = 150,

HeightRequest = 90

};

if (groupClass.SubType == ClassSubType.Other)

{

if (groupClass.WeekType != WeekType.None)

{

if (groupClass.WeekType == WeekType.Numerator)

{

sl = new StackLayout();

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(sl);

ViewSubjectGroup(sl, groupClass);

}

else

{

sl = new StackLayout();

if (sls[weekDayCounter - 1, numberCounter - 1].Children.Count == 0)

{

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(box);

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(labelSeparator);

}

else

{

sls[weekDayCounter - 1, numberCounter - 1].Children.RemoveAt(2);

}

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(sl);

ViewSubjectGroup(sl, groupClass);

}

if (sls[weekDayCounter - 1, numberCounter - 1].Children.Count == 1)

{

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(labelSeparator);

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(box);

}

}

else

{

sl = new StackLayout();

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(sl);

ViewSubjectGroup(sl, groupClass);

}

}

else

{

if (sls[weekDayCounter - 1, numberCounter - 1].Children.Count > 0)

{

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(labelSeparator);

}

sl = new StackLayout();

Label labelDate = new Label();

labelDate.Text = groupClass.Date.Value.ToShortDateString();

Label labelSubType = new Label();

if (groupClass.SubType == ClassSubType.Consultation)

{

labelSubType.Text = "Консультація";

}

else

{

labelSubType.Text = "Модульний контроль";

}

Label labelSubject = new Label();

labelSubject.Text = groupClass.Subject;

Label labelTeacher = new Label();

labelTeacher.Text = groupClass.Teacher.Name;

labelDate.FontSize = 20; labelSubject.FontAttributes = FontAttributes.Bold;

labelSubType.FontSize = 20; labelSubject.FontAttributes = FontAttributes.Bold;

labelSubject.FontSize = 18; labelSubject.FontAttributes = FontAttributes.Bold;

labelTeacher.FontSize = 18; labelTeacher.FontAttributes = FontAttributes.Italic;

sl.Children.Add(labelDate);

sl.Children.Add(labelSubType);

sl.Children.Add(labelSubject);

sl.Children.Add(labelTeacher);

sls[weekDayCounter - 1, numberCounter - 1].Children.Add(sl);

}

}

public void ViewSubjectGroup(StackLayout sl, Class groupClass)

{

Label labelSubject = new Label();

labelSubject.Text = groupClass.Subject;

Label labelTeacher = new Label();

labelTeacher.Text = groupClass.Teacher.Name;

Label labelRoom = new Label();

labelRoom.Text = "ауд." + groupClass.Auditory;

labelSubject.FontSize = 18; labelSubject.FontAttributes = FontAttributes.Bold;

labelTeacher.FontSize = 18; labelTeacher.FontAttributes = FontAttributes.Italic;

labelRoom.FontSize = 16;

sl.Children.Add(labelSubject);

sl.Children.Add(labelTeacher);

sl.Children.Add(labelRoom);

}

private async Task ViewScheduleTeacher(string teacher)

{

CleanSL(sls);

var searchCriteriaTeacher = new SearchCriteria(scheduleName, teacher, null);

var teacherClasses = await scheduleService.GetClasses(searchCriteriaTeacher);

var sortTeacherClasses = teacherClasses.OrderBy(x => x.Date).ToArray();

foreach (var teacherClass in sortTeacherClasses)

{

for (int weekDayCounter = 1; weekDayCounter <= maxWeekday; weekDayCounter++)

{

if (teacherClass.WeekDay == weekDayCounter)

{

for (int numberCounter = 1; numberCounter <= maxNumber; numberCounter++)

{

if (teacherClass.Number == numberCounter)

{

ViewSubjectsTeacher(sls, teacherClass, weekDayCounter, numberCounter);

}

}

}

}

}

}

public void ViewSubjectsTeacher(StackLayout[,] sls, Class teacherClass, int weekDayCounter, int numberCounter)

{

Label labelSeparator = new Label();

labelSeparator.Text = "----------------------------------------------";

labelSeparator.FontAttributes = FontAttributes.Bold;

StackLayout sl;

BoxView box = new BoxView

{

WidthRequest = 150,

HeightRequest = 90

};

int i = weekDayCounter - 1;

int j = numberCounter - 1;

bool isOneGroup = true;

if (sls[i, j].Children.Count > 0)

{

isOneGroup = false;

}

if (teacherClass.SubType == ClassSubType.Other)

{

if (teacherClass.WeekType != WeekType.None)

{

if (teacherClass.WeekType == WeekType.Numerator)

{

if (isOneGroup == false)

{

sls[i, j].Children.RemoveAt(sls[i, j].Children.Count - 1);

sls[i, j].Children.RemoveAt(sls[i, j].Children.Count - 1);

}

sl = new StackLayout();

sls[i, j].Children.Add(sl);

ViewSubjectTeacher(sl, teacherClass, isOneGroup);

sls[i, j].Children.Add(labelSeparator);

sls[i, j].Children.Add(box);

}

else

{

sl = new StackLayout();

if (sls[i, j].Children.Count == 0)

{

sls[i, j].Children.Add(box);

sls[i, j].Children.Add(labelSeparator);

}

else

{

if (sls[i, j].Children[sls[i, j].Children.Count - 2] is BoxView)

{

sls[i, j].Children.RemoveAt(sls[i, j].Children.Count - 2);

}

}

string str = "";

if (sls[i, j].Children[sls[i, j].Children.Count - 1] is Label label1)

{

str = label1.Text;

}

if (str == labelSeparator.Text)

{

isOneGroup = true;

}

else

{

isOneGroup = false;

}

sls[i, j].Children.Add(sl);

ViewSubjectTeacher(sl, teacherClass, isOneGroup);

}

}

else

{

sl = new StackLayout();

sls[i, j].Children.Add(sl);

ViewSubjectTeacher(sl, teacherClass, isOneGroup);

}

}

else

{

sl = new StackLayout();

Label labelDate = new Label();

labelDate.Text = teacherClass.Date.Value.ToShortDateString();

Label labelSubType = new Label();

if (teacherClass.SubType == ClassSubType.Consultation)

{

labelSubType.Text = "Консультація";

}

else

{

labelSubType.Text = "Модульний контроль";

}

Label labelSubject = new Label();

labelSubject.Text = teacherClass.Subject;

Label labelGroup = new Label();

labelGroup.Text = teacherClass.Group.Name;

labelDate.FontSize = 20; labelSubject.FontAttributes = FontAttributes.Bold;

labelSubType.FontSize = 20; labelSubject.FontAttributes = FontAttributes.Bold;

labelSubject.FontSize = 18; labelSubject.FontAttributes = FontAttributes.Bold;

labelGroup.FontSize = 18; labelGroup.FontAttributes = FontAttributes.Italic;

Label label = new Label();

if (sls[i, j].Children.Count > 0)

label = sls[i, j].Children.OfType<Label>().FirstOrDefault();

if (sls[i, j].Children.Count == 0)

{

sls[i, j].Children.Add(labelDate);

sl.Children.Add(labelSubType);

sl.Children.Add(labelSubject);

sl.Children.Add(labelGroup);

}

else if (label.Text != labelDate.Text)

{

string str = "";

if (sls[i, j].Children[sls[i, j].Children.Count - 2] is Label label1)

str = label1.Text;

if (str != labelDate.Text)

{

sls[i, j].Children.Add(labelSeparator);

sls[i, j].Children.Add(labelDate);

sl.Children.Add(labelSubType);

sl.Children.Add(labelSubject);

sl.Children.Add(labelGroup);

}

else

sl.Children.Add(labelGroup);

}

else

{

sl.Children.Add(labelGroup);

}

sls[i, j].Children.Add(sl);

}

}

public void ViewSubjectTeacher(StackLayout sl, Class teacherClass, bool isOneGroup)

{

Label labelSubject = new Label();

labelSubject.Text = teacherClass.Subject;

Label labelGroup = new Label();

labelGroup.Text = teacherClass.Group.Name;

Label labelRoom = new Label();

labelRoom.Text = "ауд." + teacherClass.Auditory;

labelSubject.FontSize = 18; labelSubject.FontAttributes = FontAttributes.Bold;

labelGroup.FontSize = 18; labelGroup.FontAttributes = FontAttributes.Italic;

labelRoom.FontSize = 16;

if (isOneGroup)

{

sl.Children.Add(labelSubject);

sl.Children.Add(labelRoom);

sl.Children.Add(labelGroup);

}

else

{

sl.Children.Add(labelGroup);

}

}

public void CleanSL(StackLayout[,] sls)

{

int weekDayCounter = 1;

while (weekDayCounter <= maxWeekday)

{

int numberCounter = 1;

while (numberCounter <= maxNumber)

{

sls[weekDayCounter - 1, numberCounter - 1].Children.Clear();

numberCounter++;

}

weekDayCounter++;

}

}

public async void ShowScheduleForGroup()

{

var groups = await scheduleService.GetGroups(scheduleName);

if (groupPicker.SelectedItem != null)

{

foreach (var groupName in groups)

{

if (groupPicker.Items[groupPicker.SelectedIndex] == groupName)

{

await ViewScheduleGroup(groupName);

break;

}

}

}

}

public async void ShowScheduleForTeacher()

{

var teachers = await scheduleService.GetTeachers(scheduleName);

if (teacherPicker.SelectedItem != null)

{

foreach (var teacherName in teachers)

{

if (teacherPicker.Items[teacherPicker.SelectedIndex] == teacherName)

{

await ViewScheduleTeacher(teacherName);

break;

}

}

}

}

public void onSelectedGroupChange(object sender, EventArgs e)

{

if (groupRadioButton.IsChecked)

{

ShowScheduleForGroup();

}

CloseDayFrame();

}

public void onSelectedTeacherChange(object sender, EventArgs e)

{

if (teacherRadioButton.IsChecked)

{

ShowScheduleForTeacher();

}

CloseDayFrame();

}

private void onMondayCheckChange(object sender, CheckedChangedEventArgs e)

{

mondayFrame.IsVisible = mondayCheckbox.IsChecked;

}

private void onTuesdayCheckChange(object sender, CheckedChangedEventArgs e)

{

tuesdayFrame.IsVisible = tuesdayCheckbox.IsChecked;

}

private void onWednesdayCheckChange(object sender, CheckedChangedEventArgs e)

{

wednesdayFrame.IsVisible = wednesdayCheckbox.IsChecked;

}

private void onThursdayCheckChange(object sender, CheckedChangedEventArgs e)

{

thursdayFrame.IsVisible = thursdayCheckbox.IsChecked;

}

private void onFridayCheckChange(object sender, CheckedChangedEventArgs e)

{

fridayFrame.IsVisible = fridayCheckbox.IsChecked;

}

private void onScheduleTypeChange(object sender, CheckedChangedEventArgs e)

{

if (teacherPicker.SelectedItem == null || groupPicker.SelectedItem == null)

{

CleanSL(sls);

}

groupPicker.IsVisible = groupRadioButton.IsChecked;

teacherPicker.IsVisible = !groupRadioButton.IsChecked;

if (groupRadioButton.IsChecked)

{

ShowScheduleForGroup();

}

else

{

ShowScheduleForTeacher();

}

CloseDayFrame();

}

private void CloseDayFrame()

{

mondayFrame.IsVisible = false;

tuesdayFrame.IsVisible = false;

wednesdayFrame.IsVisible = false;

thursdayFrame.IsVisible = false;

fridayFrame.IsVisible = false;

mondayCheckbox.IsChecked = false;

tuesdayCheckbox.IsChecked = false;

wednesdayCheckbox.IsChecked = false;

thursdayCheckbox.IsChecked = false;

fridayCheckbox.IsChecked = false;

}

}

}

* + 1. AssemblyInfo.cs

using Xamarin.Forms.Xaml;

[assembly: XamlCompilation(XamlCompilationOptions.Compile)]

* + 1. DependencyInjectionContainer.cs

using Core.Parsers;

using Core.Services;

using Domain.PersistenceInterfaces;

using Microsoft.Extensions.DependencyInjection;

using Persistence.InMemory;

using Persistence.SQLiteDb;

using System;

namespace Mobile

{

public static class DependencyInjectionContainer

{

public static IServiceCollection ConfigureServices(this IServiceCollection services, Action<IServiceCollection> addPlatformSpecificServices)

{

addPlatformSpecificServices.Invoke(services);

//services = services.AddInMemoryRepositories();

services = services.AddDbRepositories();

services.AddScoped<IScheduleProvider, SiteScheduleProvider>();

services.AddScoped<IFileScheduleParser, ClassFileParser>();

services.AddScoped<IFileScheduleParser, ModuleFileParser>();

services.AddScoped<ScheduleService>();

return services;

}

private static IServiceCollection AddInMemoryRepositories(this IServiceCollection services)

{

services.AddScoped<ITeacherRepository, InMemoryTeacherRepository>();

services.AddScoped<IGroupRepository, InMemoryGroupRepository>();

services.AddScoped<IScheduleRepository, InMemoryScheduleRepository>();

return services;

}

private static IServiceCollection AddDbRepositories(this IServiceCollection services)

{

services.AddSingleton<SQLiteDatabase>();

services.AddScoped<ITeacherRepository, SQLiteTeacherRepository>();

services.AddScoped<IGroupRepository, SQLiteGroupRepository>();

services.AddScoped<IScheduleRepository, SQLiteScheduleRepository>();

return services;

}

}

}

* + 1. Startup.cs;

using Microsoft.Extensions.DependencyInjection;

using System;

namespace Mobile

{

public static class Startup

{

public static IServiceProvider ServiceProvider { get; private set; }

public static IServiceProvider Init(Action<IServiceCollection> addPlatformSpecificServices)

{

var serviceProvider = new ServiceCollection()

.ConfigureServices(addPlatformSpecificServices)

.BuildServiceProvider();

ServiceProvider = serviceProvider;

return serviceProvider;

}

}

}

* 1. Mobile.Android
     1. AndroidDbPathProvider.cs

using Domain.PersistenceInterfaces;

using System;

namespace Mobile.Android

{

public class AndroidDbPathProvider : IDbPathProvider

{

public string Path => System.IO.Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal), "schedules.db3");

}

}

* + 1. MainActivity.cs

using Android.App;

using Android.Content.PM;

using Android.OS;

using Android.Runtime;

using Domain.PersistenceInterfaces;

using Microsoft.Extensions.DependencyInjection;

namespace Mobile.Android

{

[Activity(Label = "УДУНТ Розклад занять", Icon = "@drawable/icon", 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

{

protected override void OnCreate(Bundle savedInstanceState)

{

base.OnCreate(savedInstanceState);

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

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

LoadApplication(new App(AddServices));

}

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

{

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

base.OnRequestPermissionsResult(requestCode, permissions, grantResults);

}

private static void AddServices(IServiceCollection services)

{

services.AddSingleton<IDbPathProvider, AndroidDbPathProvider>();

}

}

}

* 1. Mobile.iOS
     1. AppDeleagte.cs

using Domain.PersistenceInterfaces;

using Foundation;

using Microsoft.Extensions.DependencyInjection;

using UIKit;

namespace Mobile.iOS

{

[Register("AppDelegate")]

public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate

{

public override bool FinishedLaunching(UIApplication app, NSDictionary options)

{

global::Xamarin.Forms.Forms.Init();

LoadApplication(new App(AddServices));

return base.FinishedLaunching(app, options);

}

private static void AddServices(IServiceCollection services)

{

services.AddSingleton<IDbPathProvider, iOSDbPathProvider>();

}

}

}

* + 1. iOSDbPathProvider.cs

using Domain.PersistenceInterfaces;

using System;

namespace Mobile.iOS

{

public class iOSDbPathProvider : IDbPathProvider

{

public string Path => System.IO.Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal), "..", "Library", "schedules.db3");

}

}

* + 1. Main.cs

using UIKit;

namespace Mobile.iOS

{

public class Application

{

private static void Main(string[] args)

{

UIApplication.Main(args, null, typeof(AppDelegate));

}

}

}