МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ

Проректор Українського

державного університету

науки і технологій

Анатолій РАДКЕВИЧ

ПРОГРАМА ДЛЯ ФОРМУВАННЯ WORD   
ДОКУМЕНТІВ ЗА ШАБЛОНАМИ

Текст програми

ЛИСТ ЗАТВЕРДЖЕННЯ

44165850.01365-01 12 01-ЛЗ

Завідувач кафедри КІТ

Вадим ГОРЯЧКІН

Керівник розробки

Олександр ЖЕВАГО

Виконавець

Вероніка БАГНЮК

Нормоконтролер

Світлана ВОЛКОВА

2024

ЗАТВЕРДЖЕНО

44165850.01365-01 12 01-ЛЗ

ІНТЕРНЕТ-МАГАЗИН

ПОБУТОВОЇ ТЕХНІКИ «TECHNICSHUB»

Текст програми

44165850.01365-01 12 01

Листів 53

2024

44165850.01365-01 12 01

2

АНОТАЦІЯ

Документ 44165850.01365-01 12 01 «Програма для формування Word документів за шаблонами. Текст програми» входить до складу програмної документації на комплекс з п'яти проєктів, що реалізовують створення Word документів за шаблонами.

У даному документі представлений текст програм. Програми в проєктах написані на мові C#. Об'єм пам'яті, що займають програми комплексу, складає 105 Мб. Конфігурація комп'ютера стандартна. Комплекс функціонує в середовищі операційних систем Windows 10 та наступних версіях.

44165850.01365-01 12 01

3

ЗМІСТ

1. Схема взаємодії проєктів програми4
2. Текст програми5
   1. Проєкт Domain5
   2. Проєкт Persistence7
   3. Проєкт Core14
   4. Проєкт Presentation17
   5. Проєкт WinFormsUI23

44165850.01365-01 12 01

4

1 CХЕМА ВЗАЄМОДІЇ ПРОЄКТІВ ПРОГРАМИ

На рис. 1 приведена схема взаємодії проєктів програми.

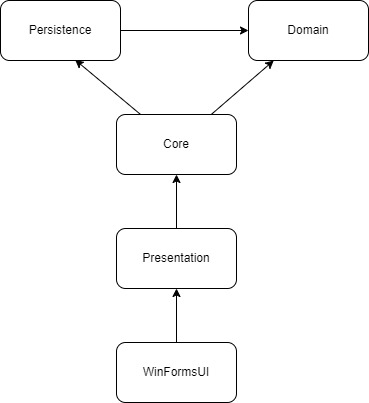


Рис.1. Схема взаємодії проєктів програми

Згідно з рис.1 взаємодія проєктів відбувається наступним чином:

* WinFormsUI взаємодіє з Presentation;
* Presentation взаємодіє з Core;
* Core взаємодіє з Persistence та Domain;
* Persistence взаємодіє з Domain.

44165850.01365-01 12 01

5

1. ТЕКСТ ПРОГРАМИ
   1. **Проєкт Domain**

Загальний опис змісту: містить моделі предметної області та інтерфейси для рівня доступу до даних. Розробник: Багнюк В.В.

Файл Template.cs

namespace Domain

{

/// <summary>

/// Шаблон

/// </summary>

public class Template

{

/// <summary>

/// Конструктор шаблона

/// </summary>

/// <param name="fileName">назва файлу</param>

/// <param name="fileContent">зміст файлу</param>

/// <exception cref="ArgumentNullException"></exception>

public Template(string fileName, string filePath, byte[] fileContent, Dictionary<string, string> fileBookmarks)

{

if (fileContent == null || !fileContent.Any())

{

throw new ArgumentNullException(nameof(fileContent));

}

this.FileName = fileName ?? throw new ArgumentNullException(nameof(fileName));

this.FileContent = fileContent;

this.FilePath = filePath ?? throw new ArgumentNullException(nameof(filePath));

this.FileBookmarks = fileBookmarks;

}

/// <summary>

/// Назва файлу

/// </summary>

public string FileName { get; }

/// <summary>

/// Шлях до файлу

/// </summary>

public string FilePath { get; }

/// <summary>

/// Зміст файлу

/// </summary>

public byte[] FileContent { get; }

/// <summary>

/// Закладки файлу

/// (ключ - назва, значення - тип)

/// </summary>

public Dictionary<string, string> FileBookmarks { get; }

}

}

44165850.01365-01 12 01

6

Файл Command.cs

namespace Domain

{

/// <summary>

/// Команда

/// </summary>

public class Command

{

/// <summary>

/// Конструктор команди

/// </summary>

/// <param name="commandName">назва команди</param>

/// <param name="commandSetting">налаштування команди</param>

/// <param name="inputTemplate">вхідний шаблон</param>

/// <param name="outputTemplate">вихідний шаблон</param>

/// <exception cref="ArgumentNullException">виключення</exception>

public Command(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate)

{

this.CommandName = commandName ?? throw new ArgumentNullException(nameof(commandName));

this.CommandSetting = commandSetting ?? throw new ArgumentNullException(nameof(commandSetting));

this.InputTemplate = inputTemplate ?? throw new ArgumentNullException(nameof(inputTemplate));

this.OutputTemplate = outputTemplate ?? throw new ArgumentNullException(nameof(outputTemplate));

}

/// <summary>

/// Налаштування команди

/// </summary>

public IReadOnlyDictionary<string, string> CommandSetting { get; }

/// <summary>

/// Назва команди

/// </summary>

public string CommandName { get; }

/// <summary>

/// Вхідий шаблон

/// </summary>

public Template InputTemplate { get; }

/// <summary>

/// Вихідний шаблон

/// </summary>

public Template OutputTemplate { get; }

}

}

44165850.01365-01 12 01

7

Файл ITemplateRepository.cs

namespace Domain

{

/// <summary>

/// Інтерфейс репозиторію шаблону

/// </summary>

public interface ITemplateRepository

{

Task<Template> Create(string fileName, string filePath, byte[] fileContent, Dictionary<string, string> fileBookmarks);

Task<IEnumerable<Template>> Get(string name);

Task Update(string oldName,

string newName,

IDictionary<string, string> newBookmarks);

Task Save(Template template);

Task Delete(string name);

}

}

Файл ICommandsRepository.cs

namespace Domain

{

/// <summary>

/// Інтерфейс репозиторію команд

/// </summary>

public interface ICommandRepository

{

Task<Command> Create(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate);

Task<IEnumerable<Command>> Get(string cmdName);

Task Update(string oldName,

string newName,

IDictionary<string, string> newSetting);

Task Save(Command command);

Task Delete(string cmdName);

}

}

* 1. **Проєкт Persistence**

Загальний опис змісту: містить реалізації інтерфейсів, описаних у проєкті Domain. Розробник: Багнюк В.В.

44165850.01365-01 12 01

8

Файл InDatabaseTemplateRepository.cs

using Domain;

using Newtonsoft.Json;

using System.Data.SQLite;

using Template = Domain.Template;

namespace Persistence

{

/// <summary>

/// Репозиторій шаблонів у базі даних

/// </summary>

public class InDatabaseTemplateRepository : ITemplateRepository

{

private SQLiteConnection connection;

/// <summary>

/// Створення підключення до

/// бази даних SQLite

/// </summary>

/// <returns>підключення до бази даних sqlite</returns>

/// <exception cref="Exception">виключення</exception>

private SQLiteConnection CreateConnection()

{

SQLiteConnection conn = new SQLiteConnection(@"Data Source=..\..\..\..\repos.db; Version = 3;");

try

{

conn.Open();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

return conn;

}

/// <summary>

/// Створення шаблону

/// (в пам'яті)

/// </summary>

/// <param name="fileName">назва файлу</param>

/// <param name="filePath">шлях до файлу</param>

/// <param name="fileContent">зміст файлу</param>

/// <param name="fileBookmarks">закладки файлу</param>

/// <returns>успішність виконання операції з результатом</returns>

/// <exception cref="ArgumentNullException">виключення</exception>

public Task<Template> Create(string fileName, string filePath, byte[] fileContent, Dictionary<string, string> fileBookmarks)

{

Template template;

if (string.IsNullOrEmpty(fileName) || fileContent == null)

throw new ArgumentNullException();

else

template = new Template(fileName, filePath, fileContent, fileBookmarks);

return Task.FromResult(template);

}

44165850.01365-01 12 01

9

/// <summary>

/// Видалення шаблону з бази даних

/// </summary>

/// <param name="name">назва шаблону</param>

/// <returns>успішність виконання операціії</returns>

public Task Delete(string name)

{

connection = CreateConnection();

var command = connection.CreateCommand();

command.CommandText = @"DELETE FROM Templates WHERE Name=@name";

command.Parameters.AddWithValue("name", name);

command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

/// <summary>

/// Отримання шаблону з бази даних

/// </summary>

/// <param name="name">назва шаблону</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Template>> Get(string name)

{

connection = CreateConnection();

var command = connection.CreateCommand();

byte[] content = new byte[0];

if (string.IsNullOrEmpty(name))

{

command.CommandText = @"SELECT \* FROM Templates";

}

else

{

command.CommandText = @"SELECT \* FROM Templates WHERE Name=@name";

command.Parameters.AddWithValue("@name", name);

}

using (var reader = command.ExecuteReader())

{

List<Template> templates = new List<Template>();

while (reader.Read())

{

var \_name = reader.GetString(1);

var path = reader.GetString(2);

content = (byte[])reader["Content"];

Dictionary<string, string> bookmarks = JsonConvert.DeserializeObject<Dictionary<string, string>>(reader.GetString(4));

templates.Add(new Template(\_name, path, content, bookmarks));

}

connection.Close();

return Task.FromResult<IEnumerable<Template>>(templates);

}

}

44165850.01365-01 12 01

10

/// <summary>

/// Збереження шаблону до бази даних

/// </summary>

/// <param name="template">шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task Save(Template template)

{

connection = CreateConnection();

string insertQuery = "INSERT INTO Templates ('Name','Path','Content','Bookmarks') VALUES(@name,@path,@content,@bookmarks)";

SQLiteCommand sqlite\_cmd = new SQLiteCommand(insertQuery, connection);

string json\_bookmarks = JsonConvert.SerializeObject(template.FileBookmarks);

sqlite\_cmd.Parameters.AddWithValue("name", template.FileName);

sqlite\_cmd.Parameters.AddWithValue("path", template.FilePath);

sqlite\_cmd.Parameters.AddWithValue("content", template.FileContent);

sqlite\_cmd.Parameters.AddWithValue("bookmarks", json\_bookmarks);

var insertResult = sqlite\_cmd.ExecuteNonQuery();

sqlite\_cmd.Connection.Close();

connection.Close();

return Task.FromResult(insertResult);

}

/// <summary>

/// Оновлення шаблону в базі данхи

/// </summary>

/// <param name="oldName">назва шаблону для оновлення</param>

/// <param name="newName">нова назва шаблону</param>

/// <param name="newBookmarks">оновлені закладки</param>

/// <returns>успішність виконання операції</returns>

public Task Update(string oldName, string newName, IDictionary<string, string> newBookmarks)

{

connection = CreateConnection();

var command = connection.CreateCommand();

string json\_bookmarks = JsonConvert.SerializeObject(newBookmarks);

command.CommandText = @"UPDATE Templates SET Name=@newname, Bookmarks=@newbookmarks WHERE Name=@oldname";

command.Parameters.AddWithValue("newname", newName);

command.Parameters.AddWithValue("newbookmarks", json\_bookmarks);

command.Parameters.AddWithValue("oldname", oldName);

\_ = command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

}

}

44165850.01365-01 12 01

11

Файл InDatabaseCommandsRepository.cs

using Domain;

using Newtonsoft.Json;

using System.Data.SQLite;

using Template = Domain.Template;

namespace Persistence

{

/// <summary>

/// Репозиторій команд у базі даних

/// </summary>

public class InDatabaseCommandsRepository : ICommandRepository

{

private SQLiteConnection connection;

/// <summary>

/// Створення підключення до

/// бази даних SQLite

/// </summary>

/// <returns>підключення до бази даних sqlite</returns>

/// <exception cref="Exception">виключення</exception>

private SQLiteConnection CreateConnection()

{

SQLiteConnection conn = new SQLiteConnection(@"Data Source=..\..\..\..\repos.db; Version = 3;");

try

{

conn.Open();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

return conn;

}

/// <summary>

/// Створення команди

/// (в пам'яті)

/// </summary>

/// <param name="commandName">назва команди</param>

/// <param name="commandSetting">налаштування команди ()</param>

/// <param name="inputTemplate">вхідний документ</param>

/// <param name="outputTemplate">вихідний шаблон</param>

/// <returns>успішність виконання операції з результатом</returns>

/// <exception cref="ArgumentNullException">виключення</exception>

public Task<Command> Create(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate)

{

Command command;

if (string.IsNullOrEmpty(commandName) || commandSetting == null || inputTemplate == null || outputTemplate == null)

throw new ArgumentNullException();

else

command = new Command(commandName, commandSetting, inputTemplate, outputTemplate);

return Task.FromResult(command);

}

44165850.01365-01 12 01

12

/// <summary>

/// Видалення команди з бази даних

/// </summary>

/// <param name="cmdName">назва команди</param>

/// <returns>успішність виконання операції</returns>

public Task Delete(string cmdName)

{

connection = CreateConnection();

var command = connection.CreateCommand();

command.CommandText = @"DELETE FROM Commands WHERE Name=@name";

command.Parameters.AddWithValue("name", cmdName);

command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

/// <summary>

/// Отримання команди з бази даних

/// </summary>

/// <param name="cmdName">назва команди</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Command>> Get(string cmdName)

{

connection = CreateConnection();

var command = connection.CreateCommand();

if (string.IsNullOrEmpty(cmdName))

{

command.CommandText = @"SELECT \* FROM Commands";

}

else

{

command.CommandText = @"SELECT \* FROM Commands WHERE Name=@name";

command.Parameters.AddWithValue("@name", cmdName);

}

using (var reader = command.ExecuteReader())

{

List<Command> commands = new List<Command>();

while (reader.Read())

{

var name = reader.GetString(1);

var input\_document = JsonConvert.DeserializeObject<Template>(reader["InputDoc"].ToString());

var output\_template = JsonConvert.DeserializeObject<Template>(reader["OutputTemplate"].ToString());

var setting\_bookmarks = JsonConvert.DeserializeObject<Dictionary<string, string>>(reader.GetString(4));

commands.Add(new Command(name, setting\_bookmarks, input\_document, output\_template));

}

connection.Close();

return Task.FromResult<IEnumerable<Command>>(commands);

}

}

44165850.01365-01 12 01

13

/// <summary>

/// Збереження команди в базі даних

/// </summary>

/// <param name="command">команда</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task Save(Command command)

{

string json\_inputdoc = JsonConvert.SerializeObject(command.InputTemplate);

string json\_outputTemplate = JsonConvert.SerializeObject(command.OutputTemplate);

string json\_setting = JsonConvert.SerializeObject(command.CommandSetting);

connection = CreateConnection();

string insertQuery = "INSERT INTO Commands ('Name','InputDoc','OutputTemplate','SettingBookmarks') " +

"VALUES(@name,@inputdoc,@outtemplate,@setting)";

SQLiteCommand sqlite\_cmd = new SQLiteCommand(insertQuery, connection);

sqlite\_cmd.Parameters.AddWithValue("@name", command.CommandName);

sqlite\_cmd.Parameters.AddWithValue("@inputdoc", json\_inputdoc);

sqlite\_cmd.Parameters.AddWithValue("@outtemplate", json\_outputTemplate);

sqlite\_cmd.Parameters.AddWithValue("@setting", json\_setting);

var insertResult = sqlite\_cmd.ExecuteNonQuery();

sqlite\_cmd.Connection.Close();

connection.Close();

return Task.FromResult(insertResult);

}

/// <summary>

/// Оновлення команди в базі даних

/// </summary>

/// <param name="oldName">назва команди для оновлення</param>

/// <param name="newName">нова назва команди</param>

/// <param name="newSetting">нові налаштування команди</param>

/// <returns>успішність виконання операції</returns>

public Task Update(string oldName, string newName, IDictionary<string, string> newSetting)

{

connection = CreateConnection();

string json\_bookmarks = JsonConvert.SerializeObject(newSetting);

var command = connection.CreateCommand();

command.CommandText = @"UPDATE Commands SET Name=@newname, SettingBookmarks=@newbookmarks WHERE Name=@oldname";

command.Parameters.AddWithValue("newname", newName);

command.Parameters.AddWithValue("newbookmarks", json\_bookmarks);

command.Parameters.AddWithValue("oldname", oldName);

\_ = command.ExecuteNonQuery();

command.Connection.Close();

connection.Close();

return Task.CompletedTask;

}

}

}

44165850.01365-01 12 01

14

* 1. **Проєкт Core.**

Загальний опис змісту: містить логіку програми (сервіси для доступу даних у базі даних). Розробник: Багнюк В.В.

Файл TemplateService.cs

using Domain;

namespace Core.Services

{ /// <summary>

/// Сервіс для шаблону

/// </summary>

public class TemplateService

{

private readonly ITemplateRepository templateRepository;

public TemplateService(ITemplateRepository templateRepository)

{

this.templateRepository = templateRepository;

}

/// <summary>

/// Отримати всі шаблони

/// </summary>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Template>> GetAllTemplates()

{

return templateRepository.Get(string.Empty);

}

/// <summary>

/// Отримати шаблон

/// </summary>

/// <param name="templateName">шаблон</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<IEnumerable<Template>> GetTemplate(string templateName)

{

return templateRepository.Get(templateName);

}

/// <summary>

/// Створити шаблон

/// </summary>

/// <param name="fileName">назва шаблону</param>

/// <param name="filePath">шлях до шаблону</param>

/// <param name="fileContent">зміст шаблону</param>

/// <returns>успішність виконання операції з результатом</returns>

public Task<Template> CreateTemplate(string fileName,

string filePath,

byte[] fileContent,

Dictionary<string, string> fileBookmarks)

{

return templateRepository.Create(fileName, filePath, fileContent, fileBookmarks);

}

44165850.01365-01 12 01

15

/// <summary>

/// Оновлення шаблону

/// </summary>

/// <param name="oldTemplate">стариий шаблон</param>

/// <param name="newTemplate">новий шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task UpdateTemplate(string oldName,

string newName,

IDictionary<string, string> newBookmarks)

{

//return templateRepository.Update(oldTemplate, newTemplate);

return templateRepository.Update(oldName, newName, newBookmarks);

}

/// <summary>

/// Видалення шаблону

/// </summary>

/// <param name="fileName">назва шаблону</param>

/// <returns>успішність виконання операції</returns>

public Task DeleteTemplate(string fileName)

{

return templateRepository.Delete(fileName);

}

/// <summary>

/// Збереження шаблону

/// </summary>

/// <param name="template">шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task Save(Template template)

{

return templateRepository.Save(template);

}

}

}

Файл CommandService.cs

using Domain;

namespace Core.Services

{

/// <summary>

/// Сервіс для команд

/// </summary>

public class CommandService

{

private readonly ICommandRepository commandRepository;

public CommandService(ICommandRepository commandRepository)

{

this.commandRepository = commandRepository;

}

/// <summary>

/// Отримати всі команди

/// </summary>

/// <returns>успішність виконання операції</returns>

public Task<IEnumerable<Command>> GetAllCommands()

{

return commandRepository.Get(string.Empty);

}

44165850.01365-01 12 01

16

/// <summary>

/// Отримати команду

/// </summary>

/// <param name="commandName">назва команди</param>

/// <returns>успішність виконання операції</returns>

public Task<IEnumerable<Command>> GetCommand(string commandName)

{

return commandRepository.Get(commandName);

}

/// <summary>

/// Створення команди

/// </summary>

/// <param name="commandName">назва команди</param>

/// <param name="commandSetting">налаштування команди</param>

/// <param name="inputTemplate">вхідний шаблон</param>

/// <param name="outputTemplate">вихідний шаблон</param>

/// <returns>успішність виконання операції</returns>

public Task<Command> CreateCommand(string commandName,

IReadOnlyDictionary<string, string> commandSetting,

Template inputTemplate,

Template outputTemplate)

{

return commandRepository.Create(commandName, commandSetting, inputTemplate, outputTemplate);

}

/// <summary>

/// Видалення команди

/// </summary>

/// <param name="commandName">назва команди</param>

/// <returns>успішність виконання операції</returns>

public Task DeleteCommand(string commandName)

{

return commandRepository.Delete(commandName);

}

/// <summary>

/// Оновлення команди

/// </summary>

/// <param name="oldCommand">стара команда</param>

/// <param name="newCommand">нова команда</param>

/// <returns>успішність виконання операції</returns>

public Task UpdateCommand(string oldName,

string newName,

IDictionary<string, string> newSetting)

{

return commandRepository.Update(oldName, newName, newSetting);

}

/// <summary>

/// Зберегти команду

/// </summary>

/// <param name="command">команда</param>

/// <returns>успішність виконання операції</returns>

public Task SaveCommand(Command command)

{

return commandRepository.Save(command);

}

}

}

44165850.01365-01 12 01

17

Файл Extensions.cs

using Domain;

using Microsoft.Extensions.DependencyInjection;

using Persistence;

namespace Core

{

/// <summary>

/// Розширення

/// </summary>

public static class Extensions

{

/// <summary>

/// Додати до пам'яті репозиторії

/// </summary>

/// <param name="services"></param>

/// <returns></returns>

public static IServiceCollection AddInMemoryRepositories(this IServiceCollection services)

{

services.AddScoped<ITemplateRepository, InDatabaseTemplateRepository>();

services.AddScoped<ICommandRepository, InDatabaseCommandsRepository>();

return services;

}

}

}

* 1. **Проєкт Presentation**

Загальний опис змісту: містить елементи партерну MVP (Model-View-Presenter). Розробник: Багнюк В.В.

Файл TemplateViewModel.cs

namespace Presentation.ViewModels

{ /// <summary>

/// Модель представлення шаблона

/// </summary>

public class TemplateViewModel

{

public string FileName { get; set; }

public string FilePath { get; set; }

public byte[] ContentFile { get; set; }

public Dictionary<string, string> BookmarksFile { get; set; }

}

}

Файл CommandViewModel.cs

namespace Presentation.ViewModels

{ /// <summary>

/// Модель представлення команди

/// </summary>

public class CommandViewModel

{

public string NameCommand { get; set; }

public TemplateViewModel InputTemplate { get; set; }

public TemplateViewModel OutputTemplate { get; set; }

public Dictionary<string, string> CommandSetting { get; set; }

44165850.01365-01 12 01

18

}

}

Файл IView.cs

namespace Presentation.Views

{ /// <summary>

/// Інтерфейс представлення

/// </summary>

public interface IView

{

void Show();

void Close();

}

}

Файл IMainView.cs

using Presentation.ViewModels;

namespace Presentation.Views

{

public interface IMainView : IView

{

public TemplateViewModel viewTemplate { get; set; }

public string viewNameTemplate { get; set; }

public event EventHandler? SaveTemplate;

public event EventHandler? DeleteTemplate;

public event EventHandler? UpdateTemplate;

public event EventHandler<TemplateViewModel> GetTemplate;

public CommandViewModel viewCommand { get; set; }

public string viewNameCommand { get; set; }

public event EventHandler? SaveCommand;

public event EventHandler? DeleteCommand;

public event EventHandler<CommandViewModel>? GetCommand;

public event EventHandler? UpdateCommand;

void SetCommandsList(IEnumerable<CommandViewModel> commands);

void SetTemplateList(IEnumerable<TemplateViewModel> templates);

void SetBookmarksDictionary(IDictionary<string, string> dictionaryBookmarks);

}

}

Файл IPresenter.cs

namespace Presentation.Presenters

{ /// <summary>

/// Інтерфейс представлення

/// </summary>

public interface IPresenter

{

Task Run();

}

}

44165850.01365-01 12 01

19

Файл IMainPresenter.cs

namespace Presentation.Presenters

{

public interface IMainPresenter : IPresenter

{

}

}

Файл MainPresenter.cs

using Core.Services;

using Presentation.ViewModels;

using Presentation.Views;

namespace Presentation.Presenters

{

public class MainPresenter : IMainPresenter

{

private readonly IMainView mainView;

private readonly CommandService commandService;

private readonly TemplateService templateService;

public MainPresenter(IMainView mainView, CommandService commandService, TemplateService templateService)

{

this.mainView = mainView;

this.commandService = commandService;

this.templateService = templateService;

this.mainView.DeleteTemplate += (s, e) => RemoveTemplate(s, e);

this.mainView.SaveTemplate += (s, e) => \_ = SaveTemplate(s, e);

this.mainView.UpdateTemplate += (s, e) => UpdateTemplate(s, e, this.mainView.viewNameTemplate, this.mainView.viewTemplate);

this.mainView.GetTemplate += (s, e) => \_ = GetTemplate(s, this.mainView.viewTemplate);

this.mainView.SaveCommand += (s, e) => \_ = SaveCommand(s, e);

this.mainView.DeleteCommand += (s, e) => DeleteCommad(s, e);

this.mainView.GetCommand += (s, e) => \_ = GetCommand(s, this.mainView.viewCommand);

this.mainView.UpdateCommand += (s, e) => UpdateCommand(s, e, this.mainView.viewNameCommand, this.mainView.viewCommand);

}

/// <summary>

/// Подія "Отримати команду"

/// </summary>

public async Task GetCommand(object sender, CommandViewModel cvm)

{

// отримати команду за назвою

var cmd = await this.commandService.GetCommand(cvm.NameCommand);

// отримати вхідний документ команди

TemplateViewModel inputTmp = new TemplateViewModel();

inputTmp.FileName = cmd.First().InputTemplate.FileName;

inputTmp.FilePath = cmd.First().InputTemplate.FilePath;

inputTmp.ContentFile = cmd.First().InputTemplate.FileContent;

inputTmp.BookmarksFile = cmd.First().InputTemplate.FileBookmarks;

// отримати вихідний шаблон команди

TemplateViewModel outputTmp = new TemplateViewModel();

outputTmp.FileName = cmd.First().OutputTemplate.FileName;

outputTmp.FilePath = cmd.First().OutputTemplate.FilePath;

outputTmp.ContentFile = cmd.First().OutputTemplate.FileContent;

44165850.01365-01 12 01

20

outputTmp.BookmarksFile = cmd.First().OutputTemplate.FileBookmarks;

cvm.InputTemplate = inputTmp;

cvm.OutputTemplate = outputTmp;

cvm.CommandSetting = (Dictionary<string, string>)cmd.First().CommandSetting;

}

/// <summary>

/// Подія "Видалити команду"

/// </summary>

public void DeleteCommad(object sender, EventArgs e)

{

this.commandService.DeleteCommand(this.mainView.viewNameCommand);

}

/// <summary>

/// Подія "Зберегти команду"

/// </summary>

public async Task SaveCommand(object sender, EventArgs e)

{

var inputTmp = await templateService.CreateTemplate(mainView.viewCommand.InputTemplate.FileName,

mainView.viewCommand.InputTemplate.FilePath,

mainView.viewCommand.InputTemplate.ContentFile,

mainView.viewCommand.InputTemplate.BookmarksFile);

var outputTmp = await templateService.GetTemplate(mainView.viewCommand.OutputTemplate.FileName);

var cmd = await commandService.CreateCommand(this.mainView.viewCommand.NameCommand, this.mainView.viewCommand.CommandSetting,

inputTmp, outputTmp.Last());

\_ = this.commandService.SaveCommand(cmd);

}

/// <summary>

/// Подія "Оновити команду"

/// </summary>

public void UpdateCommand(object sender, EventArgs e, string oldName, CommandViewModel newCommand)

{

this.commandService.UpdateCommand(oldName, newCommand.NameCommand, newCommand.CommandSetting);

}

/// <summary>

/// Подія "Отримати шаблон"

/// </summary>

public async Task GetTemplate(object sender, TemplateViewModel e)

{

var tmp = await templateService.GetTemplate(e.FileName);

e.FilePath = tmp.First().FilePath;

e.ContentFile = tmp.First().FileContent;

e.BookmarksFile = tmp.First().FileBookmarks;

}

44165850.01365-01 12 01

21

/// <summary>

/// Подія "Оновити шаблон"

/// </summary>

public void UpdateTemplate(object? sender, EventArgs e, string oldName, TemplateViewModel newTemplate)

{

this.templateService.UpdateTemplate(oldName, newTemplate.FileName, newTemplate.BookmarksFile);

}

/// <summary>

/// Подія "Видалити шаблон"

/// </summary>

public void RemoveTemplate(object sender, EventArgs e)

{

this.templateService.DeleteTemplate(this.mainView.viewNameTemplate);

}

/// <summary>

/// Подія "Зберегти шаблон"

/// </summary>

public async Task SaveTemplate(object sender, EventArgs e)

{

var template = await this.templateService.CreateTemplate(this.mainView.viewTemplate.FileName,

this.mainView.viewTemplate.FilePath, this.mainView.viewTemplate.ContentFile,

this.mainView.viewTemplate.BookmarksFile);

\_ = templateService.Save(template);

}

public async Task Run()

{

var templates = await templateService.GetAllTemplates();

if (templates.Count() != 0)

{

var viewModelTemplates = templates.Select(t => new TemplateViewModel

{

FileName = t.FileName,

FilePath = Path.GetFullPath(t.FilePath),

BookmarksFile = t.FileBookmarks,

ContentFile = t.FileContent

}).ToList();

mainView.SetTemplateList(viewModelTemplates);

}

var commands = await commandService.GetAllCommands();

if(commands.Count() != 0)

{

var viewModelCommands = commands.Select(c => new CommandViewModel

{ NameCommand = c.CommandName, // назва команди

InputTemplate = new TemplateViewModel() // вхідний документ

{ FileName = c.InputTemplate.FileName,

FilePath = c.InputTemplate.FilePath,

ContentFile = c.InputTemplate.FileContent,

BookmarksFile= c.InputTemplate.FileBookmarks,

},

OutputTemplate = new TemplateViewModel() // вихідний шаблон

{

FileName = c.OutputTemplate.FileName,

44165850.01365-01 12 01

22

FilePath = c.OutputTemplate.FilePath,

ContentFile = c.OutputTemplate.FileContent,

BookmarksFile = c.OutputTemplate.FileBookmarks,

},

CommandSetting = (Dictionary<string, string>)c.CommandSetting // налаштування

}).ToList();

mainView.SetCommandsList(viewModelCommands);

}

mainView.Show();

}

}

}

Файл Extensions.cs

using Core;

using Core.Services;

using Microsoft.Extensions.DependencyInjection;

using Presentation.Presenters;

namespace Presentation

{

public static class Extensions

{

/// <summary>

/// Використати сервіси Core

/// </summary>

/// <param name="services">інтерфейс колекції сервісів</param>

public static void UseCore(this IServiceCollection services)

{

services.AddInMemoryRepositories();

AddServices(services);

AddPresenters(services);

}

/// <summary>

/// Додати сервіс

/// </summary>

/// <param name="services">інтерфейс колекції сервісів</param>

private static void AddServices(IServiceCollection services)

{

services.AddScoped<TemplateService>();

services.AddScoped<CommandService>();

}

/// <summary>

/// Додати перезентер

/// </summary>

/// <param name="services">інтерфейс колекції сервісів<</param>

private static void AddPresenters(IServiceCollection services)

{

services.AddScoped<IMainPresenter, MainPresenter>();

}

}

}

44165850.01365-01 12 01

23

* 1. **Проєкт WinFormsUI**

Загальний опис змісту: містить інтерфейс програми у вигляді WindowsForms компонентів. Розробник: Багнюк В.В.

Файл Program.cs

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Presentation;

using Presentation.Presenters;

namespace WinFormsUI

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

ApplicationConfiguration.Initialize();

var host = CreateHostBuilder().Build();

var mainPresenter = host.Services.GetRequiredService<IMainPresenter>();

mainPresenter.Run();

Application.Run();

}

static IHostBuilder CreateHostBuilder()

{

return Host.CreateDefaultBuilder().ConfigureServices((\_, services) =>

{

services.AddViews();

services.UseCore();

});

}

}

}

Файл Extensions.cs

using Microsoft.Extensions.DependencyInjection;

using Presentation.Views;

namespace WinFormsUI

{ /// <summary>

/// Розширення

/// </summary>

public static class Extensions

{ /// <summary>

/// Додати представлення

/// </summary>

/// <param name="services"></param>

public static void AddViews(this IServiceCollection services)

{

services.AddScoped<IMainView, FormMain>();

}

}

}

44165850.01365-01 12 01

24

Файл CustomMessageBox.cs

namespace WinFormsUI

{

public static class CustomMessageBox

{

/// <summary>

/// Показати вікно повідомлення

/// </summary>

/// <param name="message">текст повідомлення</param>

/// <param name="caption">заголовок вікна</param>

/// <param name="buttons"> MessageBoxButtons кнопки (YesNo або OK)</param>

/// <returns></returns>

public static DialogResult Show(string message, string caption, MessageBoxButtons buttons)

{

DialogResult dialogResult = DialogResult.None;

switch (buttons)

{

case MessageBoxButtons.YesNo:

using (FormYesNo formYesNo = new FormYesNo())

{

formYesNo.Text = caption;

formYesNo.Message = message;

dialogResult = formYesNo.ShowDialog();

}

break;

case MessageBoxButtons.OK:

using (FormOK formOK = new FormOK())

{

formOK.Text = caption;

formOK.Message = message;

dialogResult = formOK.ShowDialog();

}

break;

}

return dialogResult;

}

}

}

Файл FormOK.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormOK : MaterialForm

{

public FormOK()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

}

public string Message

{

get { return LabelTextInfoMessage.Text; }

set { LabelTextInfoMessage.Text = value; }

}

}

}

44165850.01365-01 12 01

25

Файл FormYesNo.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormYesNo : MaterialForm

{

public FormYesNo()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

}

public string Message

{

get { return LabelTextMessage.Text; }

set { LabelTextMessage.Text = value; }

}

}

}

Файл FormTextData.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormTextData : MaterialForm

{

public string textData { get; set; }

public FormTextData()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.materialLabelMaxLengthSymbol.Text += " " + this.MultiLineTextBoxTextData.MaxLength.ToString();

}

private void MultiLineTextBoxTextData\_TextChanged(object sender, EventArgs e)

{

this.materialLabelCurrentCountSymbols.Text = this.MultiLineTextBoxTextData.Text.Length.ToString();

}

44165850.01365-01 12 01

26

private void ButtonClearTextData\_Click(object sender, EventArgs e)

{

if (this.MultiLineTextBoxTextData.Text.Length > 0)

this.MultiLineTextBoxTextData.Clear();

else

CustomMessageBox.Show("Поле для введення тексту вже очищене!", "Очищення", MessageBoxButtons.OK);

}

private void ButtonSaveTextData\_Click(object sender, EventArgs e)

{

if (this.MultiLineTextBoxTextData.Text.Length > 0)

{

textData = this.MultiLineTextBoxTextData.Text;

CustomMessageBox.Show("Введений текст успішно збережено!", "Збереження", MessageBoxButtons.OK);

}

else

{

CustomMessageBox.Show("Для збереження заповніть текстове поле!", "Збереження", MessageBoxButtons.OK);

return;

}

this.Close();

}

}

}

Файл FormImageData.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormImageData : MaterialForm

{

public Image imageData { get; set; }

public FormImageData()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

}

private void ButtonLoadImage\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

open.Filter = "Image Files(\*.jpg; \*.jpeg; \*.gif; \*.bmp)|\*.jpg; \*.jpeg; \*.gif; \*.bmp)";

if (open.ShowDialog() == DialogResult.OK)

{

pictureBoxImageData.Image = new Bitmap(open.FileName);

pictureBoxImageData.Tag = "imageData";

}

}

44165850.01365-01 12 01

27

private void ButtonSaveImage\_Click(object sender, EventArgs e)

{

if ((string)pictureBoxImageData.Tag != "imageData")

{

CustomMessageBox.Show("Завантажте зображення!", "Збереження зображення", MessageBoxButtons.OK);

return;

}

else

{

this.imageData = pictureBoxImageData.Image;

CustomMessageBox.Show("Зображення збережене успішно!", "Збереження зображення", MessageBoxButtons.OK);

}

this.Close();

}

}

}

Файл TableData.cs

namespace WinFormsUI

{

/// <summary>

/// Дані таблиці

/// (для форми FormTableData)

/// </summary>

public class TableData

{

public int numberRows { get; set; }

public int numberColumns { get; set; }

public string[,] arrayData { get; set; }

public TableData(int rows, int cols)

{

this.numberRows = rows;

this.numberColumns = cols;

this.arrayData = new string[rows, cols];

}

}

}

Файл FormTableData.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormTableData : MaterialForm

{

public TableData tableData { get; set; }

public FormTableData()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.tableData = new TableData(0, 0);

}

private void ShowTable()

{

44165850.01365-01 12 01

28

// виведення номерів стовпців таблиці і заборона сортування їх змісту

for (int i = 0; i < this.tableData.numberColumns; i++)

{

this.dataGridViewTableData.Columns[i].HeaderCell.Value = "C" + Convert.ToString(i + 1);

this.dataGridViewTableData.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;

this.dataGridViewTableData.Columns[i].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

for (int i = 0; i < this.tableData.numberRows; i++)

{

this.dataGridViewTableData.Rows[i].HeaderCell.Value = Convert.ToString(i + 1);

for (int j = 0; j < this.tableData.numberColumns; ++j)

this.dataGridViewTableData.Rows[i].Cells[j].Value = Convert.ToString(this.tableData.arrayData[i, j]);

}

}

private void ButtonGenerateTable\_Click(object sender, EventArgs e)

{

if (this.tableData.arrayData != null)

{

Array.Clear(this.tableData.arrayData, 0, this.tableData.arrayData.Length);

}

int numberRows = Convert.ToInt32(numericUpDownNumberRows.Value);

int numberColumns = Convert.ToInt32(numericUpDownNumberColumns.Value);

this.tableData = new TableData(numberRows, numberColumns);

this.dataGridViewTableData.RowCount = this.tableData.numberRows;

this.dataGridViewTableData.ColumnCount = this.tableData.numberColumns;

//заповнення таблиці

for (int i = 0; i < this.tableData.numberRows; i++)

for (int j = 0; j < this.tableData.numberColumns; j++)

this.tableData.arrayData[i, j] = "Text" + (i + j).ToString();

ShowTable();

}

private void ButtonSaveTableData\_Click(object sender, EventArgs e)

{

if (this.dataGridViewTableData.Rows.Count == 0)

{

CustomMessageBox.Show("Створіть таблицю та заповніть її даними!", "Збереження даних таблиці", MessageBoxButtons.OK);

return;

}

else

{

CustomMessageBox.Show("Дані таблиці успішно збережені!", "Збереження даних таблиці", MessageBoxButtons.OK);

}

this.Close();

}

44165850.01365-01 12 01

29

private void dataGridViewTableData\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

// індекси елемента матриці для поточной клітки таблиці

int i = this.dataGridViewTableData.CurrentCell.RowIndex;

int j = this.dataGridViewTableData.CurrentCell.ColumnIndex;

if (dataGridViewTableData.CurrentCell.Value == null)

{

CustomMessageBox.Show("Значення клітинки таблиці не має бути порожнім!", "Значення клітинок таблиці", MessageBoxButtons.OK);

}

else

{

string value = dataGridViewTableData.CurrentCell.Value.ToString() ?? throw new ArgumentNullException();

this.tableData.arrayData[i, j] = value;

}

}

private void ButtonClearTableData\_Click(object sender, EventArgs e)

{

if (this.dataGridViewTableData.Rows.Count == 0)

{

CustomMessageBox.Show("Таблиця для очищення не створена!", "Очищення", MessageBoxButtons.OK);

return;

}

for (int i = 0; i < this.tableData.numberRows; i++)

for (int j = 0; j < this.tableData.numberColumns; j++)

this.tableData.arrayData[i, j] = "";

ShowTable();

}

}

}

Файл FormEditTemplate.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormEditTemplate : MaterialForm

{

private string \_oldNameTemplate;

private Dictionary<string, string> \_newBookmarksTemplate;

private List<string> namesSavedTemplates;

public string oldNameTemplate { get => \_oldNameTemplate; set => \_oldNameTemplate = value; }

public string newNameTemplate { get => TextBoxTemplateName.Text; set => TextBoxTemplateName.Text = value; }

public Dictionary<string, string> newBookmarksTemplate { get => \_newBookmarksTemplate; set => \_newBookmarksTemplate = value; }

private bool flagSave;

public bool SavedChanges { get => flagSave; }

public FormEditTemplate(string templateName, Dictionary<string, string> templateBookmarks, List<string> namesSavedTemplates)

44165850.01365-01 12 01

30

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.TextBoxTemplateName.Text = templateName;

oldNameTemplate = TextBoxTemplateName.Text;

flagSave = false;

DataGridViewComboBoxColumn theColumn = (DataGridViewComboBoxColumn)this.dataGridViewEditTableBookmarks.Columns[2];

theColumn.Items.Add("Текст");

theColumn.Items.Add("Зображення");

theColumn.Items.Add("Таблиця");

theColumn.DefaultCellStyle.NullValue = "Текст";

foreach (var itemBookmarks in templateBookmarks)

{

this.dataGridViewEditTableBookmarks.Rows.Add(0, itemBookmarks.Key);

}

this.namesSavedTemplates = namesSavedTemplates;

}

private void dataGridViewEditTableBookmarks\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewEditTableBookmarks.Rows[e.RowIndex].Cells["EditRowNumbers"].Value = (e.RowIndex + 1).ToString();

}

private void materialButtonSaveUpdate\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(this.TextBoxTemplateName.Text)

|| Path.GetExtension(this.TextBoxTemplateName.Text) != ".docx")

{

CustomMessageBox.Show("Перевірте ведення назви шаблону! Назва шаблону "

+ this.TextBoxTemplateName.Text + " має містити.docx наприкінці.",

"Редагування шаблону", MessageBoxButtons.OK);

return;

}

else

{

foreach (var name in namesSavedTemplates)

{

if (name == this.TextBoxTemplateName.Text)

{

CustomMessageBox.Show("Назва шаблону "

+ this.TextBoxTemplateName.Text + " немає співпадати зі вже збереженими шаблонами!",

"Редагування шаблону", MessageBoxButtons.OK);

return;

}

}

}

\_newBookmarksTemplate = new Dictionary<string, string>();

44165850.01365-01 12 01

31

for (int indexRow = 0; indexRow < this.dataGridViewEditTableBookmarks.Rows.Count; indexRow++)

{

string NameBookmark = dataGridViewEditTableBookmarks.Rows[indexRow].Cells[1].FormattedValue.ToString();

string SelectedType = dataGridViewEditTableBookmarks.Rows[indexRow].Cells[2].FormattedValue.ToString();

\_newBookmarksTemplate.Add(NameBookmark, SelectedType);

}

flagSave = true;

this.Close();

}

}

}

Файл FormEditCommand.cs

using MaterialSkin;

using MaterialSkin.Controls;

namespace WinFormsUI

{

public partial class FormEditCommand : MaterialForm

{

private Dictionary<string, string> newCmdSetting;

private Dictionary<string, string> inputBookmarks;

private List<string> NamesSavedCommands;

public string newNameCommand { get => this.TextBoxNewNameCommand.Text; set => this.TextBoxNewNameCommand.Text = value; }

public Dictionary<string, string> newCommandSetting { get => newCmdSetting; set => this.newCmdSetting = value; }

private bool flagSave;

public bool SavedChanges { get => flagSave; }

public FormEditCommand(string oldCmdName, Dictionary<string, string> inputDocBookmarks, Dictionary<string, string> oldCmdSetting,

List<string> namesSavedCommands)

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.TextBoxNewNameCommand.Text = oldCmdName;

this.inputBookmarks = inputDocBookmarks;

this.flagSave = false;

this.NamesSavedCommands = namesSavedCommands;

// стовпець - вхідні закладки (документ)

DataGridViewComboBoxColumn cmbColBookmarks = (DataGridViewComboBoxColumn)this.dataGridViewEditBookmarkMatch.Columns[2];

// всі назви вхідних закладок (документ)

foreach (var inputBookmark in inputBookmarks.Keys)

{

cmbColBookmarks.Items.Add(inputBookmark);

}

44165850.01365-01 12 01

32

int j = 0;

foreach (var oldSetting in oldCmdSetting) // налаштування команди: key - вихід, value - вхід

{ // назва вибраної вхідної закладки (документ)

string defaultValue = oldSetting.Value;

this.dataGridViewEditBookmarkMatch.Rows.Add(0, oldSetting.Key); // назва вихідної закладки (шаблон)

this.dataGridViewEditBookmarkMatch.Rows[j].Cells[2].Value = defaultValue;

CmdEditInputBookmark.DefaultCellStyle.NullValue = defaultValue;

j++;

}

}

private void dataGridViewEditBookmarkMatch\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewEditBookmarkMatch.Rows[e.RowIndex].Cells["CmdEditNumberRow"].Value = (e.RowIndex + 1).ToString();

}

private void ButtonSaveChangeCommand\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(this.TextBoxNewNameCommand.Text))

{

CustomMessageBox.Show("Перевірте ведену назву команди! Поле назви команди не має бути порожнім.",

"Редагування команди", MessageBoxButtons.OK);

return;

}

else

{

foreach (var name in NamesSavedCommands)

{

if (name == this.TextBoxNewNameCommand.Text)

{

CustomMessageBox.Show("Назва команди "

+ this.TextBoxNewNameCommand.Text + " немає співпадати зі вже збереженими командами!",

"Редагування команди", MessageBoxButtons.OK);

return;

}

}

this.newCmdSetting = new Dictionary<string, string>();

for (int indexRow = 0; indexRow < this.dataGridViewEditBookmarkMatch.Rows.Count; indexRow++)

{

string outputBoookmark = this.dataGridViewEditBookmarkMatch.Rows[indexRow].Cells[1].Value.ToString();

string SelectedInputBookmark = this.dataGridViewEditBookmarkMatch.Rows[indexRow].Cells[2].FormattedValue.ToString();

newCmdSetting.Add(outputBoookmark, SelectedInputBookmark);

}

flagSave = true;

this.Close();

}

}

}

}

44165850.01365-01 12 01

33

Файл FormMain.cs

using MaterialSkin;

using MaterialSkin.Controls;

using Microsoft.WindowsAPICodePack.Dialogs;

using Presentation.ViewModels;

using Presentation.Views;

using Spire.Doc;

using Spire.Doc.Documents;

namespace WinFormsUI

{

public partial class FormMain : MaterialForm, IMainView

{

static public int IndexRowCommandTable = 0; // індекс рядка таблиці шаблонів

static public int IndexRowTemplateTable = 0; // індекс рядка таблиці команд

private DataGridViewComboBoxColumn cmbGenTypeDataBookmarks; // тип даних закладки [модуль генерації]

private TemplateViewModel cmdInputDoc; // вхідний документ [модуль команд]

private TemplateViewModel genInputTemplate; // вхідний шаблон [модуль генерації]

private Dictionary<string, object> bookmarksData { get; set; } // дані закладок [модуль генерації]

// події для шаблонів

public event EventHandler? DeleteTemplate;

public event EventHandler? SaveTemplate;

public event EventHandler? UpdateTemplate;

public event EventHandler<TemplateViewModel> GetTemplate;

// події для команд

public event EventHandler? SaveCommand;

public event EventHandler? DeleteCommand;

public event EventHandler<CommandViewModel>? GetCommand;

public event EventHandler? UpdateCommand;

// поля для представлення

private string templateName;

private string commandName;

private TemplateViewModel templateViewModel;

private CommandViewModel commandViewModel;

public string viewNameTemplate { get => templateName; set => templateName = value; }

public TemplateViewModel viewTemplate { get => templateViewModel; set => templateViewModel = value; }

public CommandViewModel viewCommand { get => commandViewModel; set => commandViewModel = value; }

public string viewNameCommand { get => commandName; set => commandName = value; }

public FormMain()

{

InitializeComponent();

var materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

this.WindowState = FormWindowState.Maximized;

// [Модуль Генерації]

// додати у стовпець таблиці закладок типи даних

44165850.01365-01 12 01

34

cmbGenTypeDataBookmarks = (DataGridViewComboBoxColumn)this.dataGridViewGenSettingBookmarks.Columns[2];

cmbGenTypeDataBookmarks.Items.Add("Текст");

cmbGenTypeDataBookmarks.Items.Add("Зображення");

cmbGenTypeDataBookmarks.Items.Add("Таблиця");

this.TextBoxCmdInputDocument.ReadOnly = true;

this.bookmarksData = new Dictionary<string, object>();

this.ComboBoxGenCommandList.Enabled = false;

// [Модуль Команд]

CmdInputBookmark = (DataGridViewComboBoxColumn)this.dataGridViewCmdBookmarkMatch.Columns[2];

// представлення вхідного документа для команди

this.cmdInputDoc = new TemplateViewModel();

}

/// <summary>

/// Задати список команд

/// </summary>

/// <param name="commands">список команд</param>

public void SetCommandsList(IEnumerable<CommandViewModel> commands)

{

// materialComboBoxCmdList.Items.Add(commands.First().Name);

this.dataGridViewTableCommand.Rows.Insert(IndexRowCommandTable, 0, commands.Last().NameCommand,

commands.Last().InputTemplate.FileName, commands.Last().OutputTemplate.FileName);

IndexRowCommandTable++;

// генератор

this.ComboBoxGenCommandList.Items.Add(commands.Last().NameCommand);

this.ComboBoxGenCommandList.SelectedIndex = 0;

}

/// <summary>

/// Задати список шабонів

/// </summary>

/// <param name="templates">список шаблонів</param>

public void SetTemplateList(IEnumerable<TemplateViewModel> templates)

{

if (this.dataGridViewTableTemplate.Rows.Count == 0) // якщо шаблони не завантажені до таблиці

{

int counter = 0; // лічильник шаблонів в пам'яті

while (templates.Count() > counter)

{

// вставити рядок з інформацією про шаблон

dataGridViewTableTemplate.Rows.Insert(IndexRowTemplateTable, 0, templates.ElementAt(counter).FileName,

templates.ElementAt(counter).FilePath);

// додати списку збережених шаблонів в модулі "Генератор"

MaterialListBoxItem listBoxItem = new MaterialListBoxItem();

listBoxItem.Text = templates.ElementAt(counter).FileName;

ListBoxGenSavedTemplates.Items.Add(listBoxItem);

ListBoxGenSavedTemplates.SelectedIndex = 0;

// вихідний шаблон до модуля команд

this.ComboBoxCmdOutputTemplate.Items.Add(templates.ElementAt(counter).FileName);

this.ComboBoxCmdOutputTemplate.SelectedIndex = 0;

44165850.01365-01 12 01

35

IndexRowTemplateTable++; // збільшення індекса рядка таблиці шаблонів

counter++;

}

}

else

{

// вставка рядка до таблиці з інформацією про шаблон

dataGridViewTableTemplate.Rows.Insert(IndexRowTemplateTable, 0, templates.Last().FileName,

templates.Last().FilePath);

IndexRowTemplateTable++;

this.ComboBoxCmdOutputTemplate.Items.Add(templates.Last().FileName);

// генератор

MaterialListBoxItem listBoxItem = new MaterialListBoxItem();

listBoxItem.Text = templates.Last().FileName;

ListBoxGenSavedTemplates.Items.Add(listBoxItem);

ListBoxGenSavedTemplates.SelectedIndex = 0;

}

}

/// <summary>

/// [Модуль шаблонів]

/// Показати закладки шаблону

/// </summary>

/// <param name="dictionaryBookmarks">словник закладок

/// (ключ: ім'я закладки, значення: тип закладки)</param>

public void SetBookmarksDictionary(IDictionary<string, string> dictionaryBookmarks)

{ // очишення та оновлення таблиці закладок

this.dataGridViewTableBookmarks.Rows.Clear();

this.dataGridViewTableBookmarks.Refresh();

foreach (var itemBookmarks in dictionaryBookmarks)

{ // додаваня рядків до таблиці закладок

this.dataGridViewTableBookmarks.Rows.Add(0, itemBookmarks.Key, itemBookmarks.Value);

}

}

/// <summary>

/// Перевірка назви шаблону

/// на коректність

/// </summary>

/// <param name="NameTemplate">назва шаблону</param>

/// <returns>true - якщо назва коректна, або false - якщо назва некоректна</returns>

public bool checkNameTemplate(string NameTemplate)

{

if (string.IsNullOrEmpty(NameTemplate) || Path.GetExtension(NameTemplate) != ".docx")

{

return false;

}

return true;

}

// перевірка чи знаходиться початок і кінець закладки

// в одному TextBody (для таблиць)

private bool checkIsSameTextBody(Document doc)

{

foreach (Bookmark bookmark in doc.Bookmarks)

44165850.01365-01 12 01

36

{

var start = bookmark.BookmarkStart.OwnerParagraph.OwnerTextBody;

var end = bookmark.BookmarkEnd.OwnerParagraph.OwnerTextBody;

if (!object.Equals(start, end))

return false;

}

return true;

}

/// <summary>

/// Отримати словник закладок з документу

/// (ключ - назва закладки, значення - тип даних (текст за замовчуванням))

/// </summary>

/// <param name="pathFile">шлях до файлу</param>

/// <returns></returns>

private Dictionary<string, string> GetBookmarksFromDoc(string pathFile)

{

Document doc = new Document(pathFile);

if (doc.Bookmarks.Count == 0)

{

CustomMessageBox.Show("Шаблон не містить закладок! Додайте закладки до шаблону.", "Повідомлення", MessageBoxButtons.OK);

return null;

}

// словник закладок

Dictionary<string, string> dictionaryBookmarks = new Dictionary<string, string>();

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

{

dictionaryBookmarks.Add(doc.Bookmarks[i].Name, "Текст");

}

return dictionaryBookmarks;

}

/// <summary>

/// [Модуль шаблонів]

/// Видалення шаблону

/// </summary>

private void dataGridViewTableTemplate\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{ // назва стовпця таблиці шаблонів

string columnTableTemplate = dataGridViewTableTemplate.Columns[e.ColumnIndex].Name;

if (columnTableTemplate == "RemoveTemplate")

{

if (this.dataGridViewTableTemplate.Rows[e.RowIndex].Cells[0].Value == null)

{

CustomMessageBox.Show("Спочатку додайте шаблон до поточного рядка або виберіть інший зі списку.",

"Видалення шаблону", MessageBoxButtons.OK);

return;

}

DialogResult dialogResult = CustomMessageBox.Show("Ви впевнені, що хочете видалити шаблон? Видалення скасувати неможливо.",

"Видалення шаблону", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes) // якщо видалити шаблон

44165850.01365-01 12 01

37

{

if (dataGridViewTableTemplate.SelectedRows.Count == 0) // якщо шаблон не вибрано

{

CustomMessageBox.Show("Для видалення виберіть шаблон зі списку.", "Видалення шаблону", MessageBoxButtons.OK);

}

else

{ // якщо шаблон вибрано

int index = dataGridViewTableTemplate.CurrentCell.RowIndex;

templateName = dataGridViewTableTemplate.Rows[index].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

DeleteTemplate?.Invoke(sender, e);

this.dataGridViewTableTemplate.Rows.RemoveAt(index);

IndexRowTemplateTable--;

if (this.dataGridViewTableBookmarks.Rows.Count > 0)

{

this.dataGridViewTableBookmarks.Rows.Clear();

this.dataGridViewTableBookmarks.Refresh();

}

this.ListBoxGenSavedTemplates.Items.RemoveAt(index);

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.dataGridViewGenSettingBookmarks.Refresh();

this.ComboBoxCmdOutputTemplate.Items.RemoveAt(index);

}

}

}

}

/// <summary>

/// [Модуль шаблонів]

/// Додавання шаблонів

/// </summary>

private void ButtonAddTemplate\_Click(object sender, EventArgs e)

{

// відкрити діалогове вікно з вибором файлу .docx

OpenFileDialog ofd = new OpenFileDialog

{

Filter = "Word|\*.docx"

};

if (ofd.ShowDialog() == DialogResult.OK)

{

FileInfo fileInfo = new FileInfo(ofd.FileName); // назва файлу

foreach (DataGridViewRow row in this.dataGridViewTableTemplate.Rows)

{

string columnValue = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException(); ;

if (columnValue == fileInfo.Name)

{

CustomMessageBox.Show("Шаблон має бути з унікальним іменем.", "Повідомлення", MessageBoxButtons.OK);

return;

}

}

templateViewModel = new TemplateViewModel();

44165850.01365-01 12 01

38

templateViewModel.FileName = fileInfo.Name;

templateViewModel.FilePath = fileInfo.FullName;

templateViewModel.ContentFile = File.ReadAllBytes(fileInfo.FullName);

templateViewModel.BookmarksFile = GetBookmarksFromDoc(fileInfo.FullName);

// збереження шаблона в пам'яті

SaveTemplate?.Invoke(sender, e);

// додати шаблон до списку для показу

List<TemplateViewModel> templates = new List<TemplateViewModel>

{

templateViewModel // Template

};

this.SetTemplateList(templates);

}

}

/// <summary>

/// [Модуль шаблонів]

/// Редагування шаблона

/// </summary>

private void ButtonEditTemplate\_Click(object sender, EventArgs e)

{

if (dataGridViewTableTemplate.SelectedRows.Count == 0)

{

CustomMessageBox.Show("Для редагування виберіть шаблон зі списку.", "Повідомлення", MessageBoxButtons.OK);

}

else

{

int indexTmp = dataGridViewTableTemplate.CurrentCell.RowIndex; // індекс вибраного шаблону

// назва шаблона та шлях до файлу

string templateName = this.dataGridViewTableTemplate.Rows[indexTmp].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

string templatePath = this.dataGridViewTableTemplate.Rows[indexTmp].Cells[2].Value.ToString() ?? throw new ArgumentNullException();

List<string> namesTemplates = new List<string>();

foreach (DataGridViewRow row in this.dataGridViewTableTemplate.Rows)

{

string name = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException();

if (name != templateName)

namesTemplates.Add(name);

}

// форма для редагування

FormEditTemplate formEditTemplate = new FormEditTemplate(templateName,

this.GetBookmarksFromDoc(templatePath), namesTemplates);

formEditTemplate.ShowDialog();

if (!formEditTemplate.SavedChanges)

return;

this.templateName = formEditTemplate.oldNameTemplate;

templateViewModel = new TemplateViewModel();

templateViewModel.FileName = formEditTemplate.newNameTemplate;

templateViewModel.BookmarksFile = formEditTemplate.newBookmarksTemplate;

// оновлення шаблону в пам'яті

UpdateTemplate?.Invoke(sender, e);

44165850.01365-01 12 01

39

this.dataGridViewTableTemplate.Rows[indexTmp].Cells[1].Value = formEditTemplate.newNameTemplate;

this.SetBookmarksDictionary(formEditTemplate.newBookmarksTemplate);

this.ListBoxGenSavedTemplates.Items[indexTmp].Text = formEditTemplate.newNameTemplate;

this.ComboBoxCmdOutputTemplate.Items[indexTmp] = formEditTemplate.newNameTemplate;

}

}

/// <summary>

/// [Модуль шаблонів]

/// Читання шаблонів (відображення закладок)

/// </summary>

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

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

private void ButtonReadTemplate\_Click(object sender, EventArgs e)

{

if (dataGridViewTableTemplate.SelectedRows.Count == 0)

{

CustomMessageBox.Show("Для редагування виберіть шаблон зі списку.", "Повідомлення", MessageBoxButtons.OK);

}

else

{

int index = dataGridViewTableTemplate.CurrentCell.RowIndex;

string nameTemplate = this.dataGridViewTableTemplate.Rows[index].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

this.templateViewModel = new TemplateViewModel();

templateViewModel.FileName = nameTemplate;

GetTemplate?.Invoke(sender, viewTemplate);

if (viewTemplate != null)

this.SetBookmarksDictionary(viewTemplate.BookmarksFile);

else

CustomMessageBox.Show("Шаблон не знайдено.", "Повідомлення", MessageBoxButtons.OK);

}

}

/// <summary>

/// [Модуль шаблонів]

/// Пошук шаблона за назвою

/// </summary>

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

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

private void TextBoxSearchTemplate\_TrailingIconClick(object sender, EventArgs e)

{

string nameTemplate = this.TextBoxSearchTemplate.Text;

if (!checkNameTemplate(nameTemplate))

{

CustomMessageBox.Show("Перевірте ведення назви шаблону! Назва шаблону " + nameTemplate + " має містити .docx наприкінці.",

"Пошук шаблону", MessageBoxButtons.OK);

return;

}

else

44165850.01365-01 12 01

40

{

if (this.dataGridViewTableTemplate.Rows.Count == 0)

{ // якщо доданих шаблонів немає

CustomMessageBox.Show("Шаблони для пошуку відсутні! Додайте шаблони.", "Пошук шаблону", MessageBoxButtons.OK);

return;

}

int rowCount = dataGridViewTableTemplate.RowCount - 1;

int i = 0; // лічильник рядків

while (i <= rowCount)

{

if (dataGridViewTableTemplate.Rows[i].Cells["NameFile"].Value.ToString() == nameTemplate)

{ // виділити знайдений шаблон

dataGridViewTableTemplate.CurrentCell = dataGridViewTableTemplate.Rows[i].Cells[0];

return;

}

i++;

}

CustomMessageBox.Show("Шаблон з назвою " + nameTemplate + " НЕ знайдено!", "Пошук шаблону", MessageBoxButtons.OK);

}

}

private void dataGridViewTableTemplate\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{ // нумерація рядків таблиці шаблонів

this.dataGridViewTableTemplate.Rows[e.RowIndex].Cells["NumberRows"].Value = (e.RowIndex + 1).ToString();

}

private void dataGridViewTableBookmarks\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{ // нумерація рядків таблиці закладок

this.dataGridViewTableBookmarks.Rows[e.RowIndex].Cells["ColumnNumber"].Value = (e.RowIndex + 1).ToString();

}

/// <summary>

/// [Модуль генерації]

/// Вибір шляху для збереження

/// сформованого документа

/// </summary>

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

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

private void ButtonSelectPathForSaved\_Click(object sender, EventArgs e)

{

CommonOpenFileDialog dialog = new CommonOpenFileDialog();

dialog.InitialDirectory = "C:\\Users";

dialog.IsFolderPicker = true;

if (dialog.ShowDialog() == CommonFileDialogResult.Ok)

{

this.TextBoxGenPathSaveDocument.Text = dialog.FileName;

}

}

private void dataGridViewGenSettingBookmarks\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

44165850.01365-01 12 01

41

{

this.dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells["GenNumberRows"].Value = (e.RowIndex + 1).ToString();

}

/// <summary>

/// [Модуль генерації]

/// Завантаження шаблона

/// </summary>

private void TextBoxGenUploadTemplate\_TrailingIconClick(object sender, EventArgs e)

{

// відкрити діалогове вікно з вибором файлу .docx

OpenFileDialog ofd = new OpenFileDialog

{

Filter = "Word|\*.docx"

};

if (ofd.ShowDialog() == DialogResult.OK)

{

// інформація про файл

FileInfo fileInfo = new FileInfo(ofd.FileName);

if (ListBoxGenSavedTemplates.Items.Count > 0)

{

MaterialListBoxItem item = new MaterialListBoxItem();

item.Text = ofd.FileName;

var isContains = ListBoxGenSavedTemplates.Items.Contains(item);

if (isContains)

{

CustomMessageBox.Show("Назва шаблона не має співпадати з вже завантаженим шаблоном! Змініть назву одного з файлів.",

"Завантаження шаблона", MessageBoxButtons.OK);

return;

}

}

this.TextBoxGenUploadTemplate.Text = fileInfo.Name;

Document doc = new Document(fileInfo.FullName);

if (doc.Bookmarks.Count == 0)

{ // якщо в документі немає закладок

CustomMessageBox.Show("Шаблон не містить закладок! Додайте закладки до шаблону.",

"Завантаження шаблона", MessageBoxButtons.OK);

return;

}

// словник закладок

Dictionary<string, string> dictionaryBookmarks = new Dictionary<string, string>();

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

{

dictionaryBookmarks.Add(doc.Bookmarks[i].Name, "Текст");

}

genInputTemplate = new TemplateViewModel();

genInputTemplate.FileName = fileInfo.Name;

genInputTemplate.BookmarksFile = dictionaryBookmarks;

genInputTemplate.FilePath = fileInfo.FullName;

genInputTemplate.ContentFile = File.ReadAllBytes(fileInfo.FullName);

44165850.01365-01 12 01

42

if (this.bookmarksData.Count > 0)

this.bookmarksData.Clear();

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.dataGridViewGenSettingBookmarks.Refresh();

int indexRow = 0;

foreach (var item in genInputTemplate.BookmarksFile)

{

string val = item.Value;

this.dataGridViewGenSettingBookmarks.Rows.Add(0, item.Key);

this.dataGridViewGenSettingBookmarks.Rows[indexRow].Cells[2].Value = val;

cmbGenTypeDataBookmarks.DefaultCellStyle.NullValue = val;

indexRow++;

}

}

}

/// <summary>

/// [Модуль генерації]

/// Задання даних для закладок шаблона

/// залежно від типу (текст, зображення, таблиця)

/// </summary>

private void dataGridViewGenSettingBookmarks\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

string columnTableBookmark = this.dataGridViewGenSettingBookmarks.Columns[e.ColumnIndex].Name;

if (columnTableBookmark == "GenEnterData")

{

string typeData = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[2].FormattedValue.ToString()

?? throw new ArgumentNullException();

if (typeData == "Текст")

{

string nameBookmark = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[1].FormattedValue.ToString()

?? throw new ArgumentNullException();

FormTextData formTextData = new FormTextData();

formTextData.ShowDialog();

if (this.bookmarksData.ContainsKey(nameBookmark))

{

this.bookmarksData[nameBookmark] = formTextData.textData;

} else

this.bookmarksData.Add(nameBookmark, formTextData.textData);

}

else if (typeData == "Зображення")

{

string nameBookmark = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[1].FormattedValue.ToString()

?? throw new ArgumentNullException();

FormImageData formImageData = new FormImageData();

formImageData.ShowDialog();

if (this.bookmarksData.ContainsKey(nameBookmark))

{

this.bookmarksData[nameBookmark] = formImageData.imageData;

} else

this.bookmarksData.Add(nameBookmark, formImageData.imageData);

}

44165850.01365-01 12 01

43

else if (typeData == "Таблиця")

{

string nameBookmark = dataGridViewGenSettingBookmarks.Rows[e.RowIndex].Cells[1].FormattedValue.ToString()

?? throw new ArgumentNullException(); ;

FormTableData formTableData = new FormTableData();

formTableData.ShowDialog();

if (this.bookmarksData.ContainsKey(nameBookmark))

{

this.bookmarksData[nameBookmark] = formTableData.tableData;

} else

this.bookmarksData.Add(nameBookmark, formTableData.tableData);

}

}

}

/// <summary>

/// [Модуль генерації]

/// Створення таблиці для

/// вставки у Word документ

/// </summary>

/// <param name="document">документ</param>

/// <param name="tableInsert">дані таблиці для заповнення</param>

/// <returns>таблиця</returns>

Table CreateTableForWord(Document document, TableData tableInsert)

{

Table table = new Table(document, true);

table.ResetCells(tableInsert.numberRows, tableInsert.numberColumns);

for (int i = 0; i < tableInsert.numberRows; i++)

{

TableRow row = table.Rows[i];

for (int j = 0; j < tableInsert.numberColumns; j++)

{

TableCell cell = row.Cells[j];

Paragraph para = cell.AddParagraph();

para.AppendText(tableInsert.arrayData[i, j]);

}

}

return table;

}

/// <summary>

/// [Модуль генерації]

/// Створення документа на основі шаблона

/// </summary>

private void ButtonGenerateDocument\_Click(object sender, EventArgs e)

{

string pathFile = this.TextBoxGenPathSaveDocument.Text.ToString(); // шлях для збрежеження

string nameFile = this.TextBoxGenNameOutputDocument.Text; // назва вихідного файлу

if (string.IsNullOrEmpty(pathFile) || (checkNameTemplate(nameFile) == false)

|| (this.dataGridViewGenSettingBookmarks.Rows.Count == 0))

{

CustomMessageBox.Show("Документ не сформовано! Можливо, не вказаний шлях для збереження, " +

"або некорекна назва вихідного документа чи не показані закладки шаблона.", "Формування документа", MessageBoxButtons.OK);

return;

44165850.01365-01 12 01

44

}

else

{

if (SwitchGenUseCommand.Checked)

{ // формування документа з використанням даних команди

int indexSelectedCommand = ComboBoxGenCommandList.SelectedIndex;

string nameSelectedCommand = ComboBoxGenCommandList.Items[indexSelectedCommand].ToString()

?? throw new ArgumentNullException();

this.commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = nameSelectedCommand;

this.GetCommand?.Invoke(sender, viewCommand);

CustomMessageBox.Show("Назва: " + viewCommand.NameCommand + ", Вхідний документ: " +

viewCommand.InputTemplate.FileName + ", Вихідний шаблон: " + viewCommand.OutputTemplate.FileName,

"Вибрана команда", MessageBoxButtons.OK);

// вхідний документ

Document inputDocument = new Document();

MemoryStream streamInput = new MemoryStream(viewCommand.InputTemplate.ContentFile);

inputDocument.LoadFromStream(streamInput, FileFormat.Docx);

// вихідний шаблон

Document outputTemplate = new Document();

MemoryStream streamOutput = new MemoryStream(viewCommand.OutputTemplate.ContentFile);

outputTemplate.LoadFromStream(streamOutput, FileFormat.Docx);

// навігатори по закладкам вхідного документа та вихідного шаблона відповідно

BookmarksNavigator navigatorInput = new BookmarksNavigator(inputDocument);

BookmarksNavigator navigatorOutput = new BookmarksNavigator(outputTemplate);

foreach (var setting in viewCommand.CommandSetting)

{

navigatorInput.MoveToBookmark(setting.Value); // до вхідної закладки

TextBodyPart inTextBodyPart = navigatorInput.GetBookmarkContent();

navigatorOutput.MoveToBookmark(setting.Key); // до вихідної закладки

navigatorOutput.ReplaceBookmarkContent(inTextBodyPart); // замінити

}

string fullpath = pathFile + @"\" + nameFile;

outputTemplate.SaveToFile(fullpath, FileFormat.Docx);

CustomMessageBox.Show("Документ успішно створено! Шлях: " + fullpath,

"Формування документа", MessageBoxButtons.OK);

}

else

{

44165850.01365-01 12 01

45

Document inputTemplate = new Document();

if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenSaveTemplate)

{

//CustomMessageBox.Show("Ви вибрали шаблон зі списку збережених шаблонів", "Вибір шаблона", MessageBoxButtons.OK);

if (this.bookmarksData.Count != this.dataGridViewGenSettingBookmarks.Rows.Count)

{

CustomMessageBox.Show("Задайте дані для всіх закладок шаблона!",

"Формування документа", MessageBoxButtons.OK);

return;

}

int indexSelectedTemplate = ListBoxGenSavedTemplates.SelectedIndex;

this.templateViewModel = new TemplateViewModel();

templateViewModel.FileName = ListBoxGenSavedTemplates.Items[indexSelectedTemplate].Text;

this.GetTemplate.Invoke(sender, viewTemplate);

// bookmarksData: key - закладка, value - дані

MemoryStream streamInput = new MemoryStream(viewTemplate.ContentFile);

inputTemplate.LoadFromStream(streamInput, FileFormat.Docx);

}

else if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenLoadTemplate)

{

// CustomMessageBox.Show("Ви вибрали завантажений шаблон", "Вибір шаблона", MessageBoxButtons.OK);

if (string.IsNullOrEmpty(genInputTemplate.FileName))

{

CustomMessageBox.Show("Завантажте шаблон або виберіть зі списку збережених!",

"Вибір шаблона", MessageBoxButtons.OK);

return;

}

else if (this.bookmarksData.Count != this.dataGridViewGenSettingBookmarks.Rows.Count)

{

CustomMessageBox.Show("Задайте дані для всіх закладок шаблона!",

"Формування документа", MessageBoxButtons.OK);

return;

}

MemoryStream streamInput = new MemoryStream(genInputTemplate.ContentFile);

inputTemplate.LoadFromStream(streamInput, FileFormat.Docx);

}

BookmarksNavigator navigatorInput = new BookmarksNavigator(inputTemplate);

foreach (var item in bookmarksData)

{

navigatorInput.MoveToBookmark(item.Key);

44165850.01365-01 12 01

46

if (item.Value.GetType() == typeof(string)) // якщо дані текст

{

if (navigatorInput.GetBookmarkContent() != null)

{

navigatorInput.ReplaceBookmarkContent(item.Value.ToString(), true);

}

else

navigatorInput.InsertText(item.Value.ToString());

}

else if (item.Value.GetType() == typeof(System.Drawing.Bitmap)) // якщо дані зображення

{

Section section0 = inputTemplate.AddSection();

Paragraph paragraph = section0.AddParagraph();

paragraph.AppendPicture((System.Drawing.Image)item.Value);

// navigatorInput.InsertParagraph(paragraph);

TextBodyPart textBodyPart = new TextBodyPart(inputTemplate);

textBodyPart.BodyItems.Add(paragraph);

navigatorInput.ReplaceBookmarkContent(textBodyPart);

}

else if (item.Value.GetType() == typeof(TableData)) // якщо дані таблиця

{

Table table = CreateTableForWord(inputTemplate, (TableData)item.Value);

TextBodyPart part = new TextBodyPart(inputTemplate);

part.BodyItems.Add(table);

// navigatorInput.InsertTextBodyPart(part);

navigatorInput.ReplaceBookmarkContent(part);

}

}

string fullpath = pathFile + @"\" + nameFile;

inputTemplate.SaveToFile(fullpath, FileFormat.Docx);

CustomMessageBox.Show("Документ успішно створено! Шлях: " + fullpath,

"Формування документа", MessageBoxButtons.OK);

}

}

}

/// <summary>

/// [Модуль генерації]

/// TabControl вибору шаблона

/// серед збережених або завантажити

/// </summary>

private void TabControlGenSelectTemplate\_SelectedIndexChanged(object sender, EventArgs e)

{

if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenSaveTemplate)

{

if (this.dataGridViewTableTemplate.Rows.Count > 0)

{

this.dataGridViewGenSettingBookmarks.Refresh();

this.ListBoxGenSavedTemplates\_SelectedValueChanged(sender, null);

44165850.01365-01 12 01

47

}

}

else if (materialTabControlGenSelectTemplate.SelectedTab == tabPageGenLoadTemplate)

{

if (bookmarksData.Count > 0)

{

var result = CustomMessageBox.Show("Увага! Попередньо задані дані для закладок поточного шаблону буде видалено!" +

"Продовжити?", "Попередження", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

this.bookmarksData.Clear();

else

return;

}

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.TextBoxGenUploadTemplate.Text = string.Empty;

}

}

/// <summary>

/// [Модуль генерації]

/// Список збережених шаблонів

/// </summary>

private void ListBoxGenSavedTemplates\_SelectedValueChanged(object sender, MaterialListBoxItem selectedItem)

{

int indexSelectedTemplate = ListBoxGenSavedTemplates.SelectedIndex;

if (bookmarksData.Count > 0)

{

var result = CustomMessageBox.Show("Увага! Попередньо задані дані для закладок поточного шаблону буде видалено!" +

" Продовжити?", "Попередження", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

this.bookmarksData.Clear();

}

else

{

if (indexSelectedTemplate > 0)

ListBoxGenSavedTemplates.SelectedIndex = indexSelectedTemplate - 1;

return;

}

}

this.dataGridViewGenSettingBookmarks.Rows.Clear();

this.dataGridViewGenSettingBookmarks.Refresh();

this.templateViewModel = new TemplateViewModel();

templateViewModel.FileName = ListBoxGenSavedTemplates.Items[indexSelectedTemplate].Text;

// отримати шаблон з бази даних

this.GetTemplate.Invoke(sender, viewTemplate);

int indexRow = 0;

foreach (var item in viewTemplate.BookmarksFile)

{

string val = item.Value.ToString();

this.dataGridViewGenSettingBookmarks.Rows.Add(0, item.Key);

44165850.01365-01 12 01

48

this.dataGridViewGenSettingBookmarks.Rows[indexRow].Cells[2].Value = val;

cmbGenTypeDataBookmarks.DefaultCellStyle.NullValue = val;

indexRow++;44165850.01365-01 12 01

}

}

/// <summary>

/// [Модуль команд]

/// завантажити вхідний документ

/// </summary>

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

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

private void TextBoxCmdInputDocument\_TrailingIconClick(object sender, EventArgs e)

{

// відкрити діалогове вікно з вибором файлу .docx

OpenFileDialog ofd = new OpenFileDialog

{

Filter = "Word|\*.docx"

};

if (ofd.ShowDialog() == DialogResult.OK)

{

// інформація про файл

FileInfo fileInfo = new FileInfo(ofd.FileName);

if (ComboBoxCmdOutputTemplate.Items.Count > 0)

{

var isContains = ComboBoxCmdOutputTemplate.Items.Contains(ofd.FileName);

if (isContains)

{

CustomMessageBox.Show("Назва документа не має співпадати з вже завантаженим шаблоном! Змініть назву одного з файлів.",

"Завантаження документа", MessageBoxButtons.OK);

return;

}

}

this.TextBoxCmdInputDocument.Text = fileInfo.Name;

Document doc = new Document(fileInfo.FullName);

if (doc.Bookmarks.Count == 0)

{ // якщо в документі немає закладок

CustomMessageBox.Show("Шаблон не містить закладок! Додайте закладки до шаблону.", "Завантаження документа", MessageBoxButtons.OK);

return;

}

if (!checkIsSameTextBody(doc))

{

CustomMessageBox.Show("Початок і кінець закладки має бути в одному текстовому тілі! " +

"Перевірте закладку до таблиці: початок має бути на один рядок вище таблиці, а кінець - нижче.",

"Завантаження документа",

MessageBoxButtons.OK);

this.TextBoxCmdInputDocument.Text = "";

return;

}

44165850.01365-01 12 01

49

// словник закладок

Dictionary<string, string> dictionaryBookmarks = new Dictionary<string, string>();

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

{

dictionaryBookmarks.Add(doc.Bookmarks[i].Name, "Текст");

}

cmdInputDoc.FileName = fileInfo.Name;

cmdInputDoc.BookmarksFile = dictionaryBookmarks;

cmdInputDoc.FilePath = fileInfo.FullName;

cmdInputDoc.ContentFile = File.ReadAllBytes(fileInfo.FullName);

}

}

/// <summary>

/// [Модуль команд]

/// Показ закладок (налаштувань команди)

/// </summary>

private void ButtonCmdViewBookmark\_Click(object sender, EventArgs e)

{

if (this.dataGridViewCmdBookmarkMatch.Rows.Count > 0)

{

DialogResult dialogResult = CustomMessageBox.Show("Закладки вже показані." +

" Якщо ви не створили з наявними параметрами конвертування команду, то вони будуть видалені. Продовжити?",

"Показ закладок", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes)

{

CmdInputBookmark.Items.Clear();

this.dataGridViewCmdBookmarkMatch.Rows.Clear();

this.dataGridViewCmdBookmarkMatch.Refresh();

}

else

return;

}

else

{

int indexOutputTemplate = ComboBoxCmdOutputTemplate.SelectedIndex;

if (TextBoxCmdInputDocument.Text.Length == 0)

{

CustomMessageBox.Show("Спочатку завантажте заповнений шаблон!", "Показ закладок", MessageBoxButtons.OK);

return;

}

this.templateViewModel = new TemplateViewModel();

string nameSelectedTemplate = ComboBoxCmdOutputTemplate.Items[indexOutputTemplate].ToString()

?? throw new ArgumentNullException();

templateViewModel.FileName = nameSelectedTemplate;

GetTemplate?.Invoke(sender, viewTemplate);

int countOutput = viewTemplate.BookmarksFile.Count;

int countInput = cmdInputDoc.BookmarksFile.Count();

if (countOutput != countInput)

{

CustomMessageBox.Show("Кількість закладок у вхідному документі та вихідному шаблоні має бути однакова! " +

"Вхідний документ має " + countInput.ToString() + " закладки(ок), а вихідний шаблон " + countOutput.ToString(),

44165850.01365-01 12 01

50

"Показ закладок", MessageBoxButtons.OK);

return;

}

DataGridViewComboBoxColumn inputBookmarks = (DataGridViewComboBoxColumn)this.dataGridViewCmdBookmarkMatch.Columns[2];

foreach (var input in cmdInputDoc.BookmarksFile.Keys)

{

inputBookmarks.Items.Add(input);

inputBookmarks.DefaultCellStyle.NullValue = inputBookmarks.Items[0].ToString();

}

foreach (var output in viewTemplate.BookmarksFile.Keys)

{

this.dataGridViewCmdBookmarkMatch.Rows.Add(0, output);

}

}

}

private void dataGridViewCmdBookmarkMatch\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewCmdBookmarkMatch.Rows[e.RowIndex].Cells["CmdNumberRow"].Value = (e.RowIndex + 1).ToString();

}

private void dataGridViewTableCommand\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

this.dataGridViewTableCommand.Rows[e.RowIndex].Cells["CmdNumberRowCommand"].Value = (e.RowIndex + 1).ToString();

}

/// <summary>

/// [Модуль команд]

/// Створення команди

/// </summary>

private void ButtonCreateCommand\_Click(object sender, EventArgs e)

{

string commandName = this.TextBoxCmdCommandName.Text;

int indexOutputTemplate = ComboBoxCmdOutputTemplate.SelectedIndex;

if(this.dataGridViewCmdBookmarkMatch.Rows.Count == 0) {

CustomMessageBox.Show("Задайте налаштування команди!", "Створення команди", MessageBoxButtons.OK);

return;

}

if(this.dataGridViewTableBookmarks.Rows.Count > 0)

{

foreach (DataGridViewRow row in this.dataGridViewTableCommand.Rows)

{

string name = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException();

if (commandName == name)

{

CustomMessageBox.Show("Команда з такою назвою вже існує!", "Створення команди", MessageBoxButtons.OK);

return;

}

44165850.01365-01 12 01

51

}

}

Dictionary<string, string> cmdSetting = new Dictionary<string, string>();

for (int indexRow = 0; indexRow < this.dataGridViewCmdBookmarkMatch.Rows.Count; indexRow++)

{

string OutputBookmark = this.dataGridViewCmdBookmarkMatch.Rows[indexRow].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

string InputBookmark = this.dataGridViewCmdBookmarkMatch.Rows[indexRow].Cells[2].FormattedValue.ToString() ?? throw new ArgumentNullException();

if (string.Equals(OutputBookmark, InputBookmark))

{

CustomMessageBox.Show("Назви вхідної та вихідної закладки не мають співпадати! Або виберіть закладку з іншою назвою, " +

" або змініть назву закладки в " + this.TextBoxCmdInputDocument.Text + " та завантажте його знову.",

"Створення команди", MessageBoxButtons.OK);

return;

}

cmdSetting.Add(OutputBookmark, InputBookmark);

}

this.templateViewModel = new TemplateViewModel();

string nameSelectedTemplate = ComboBoxCmdOutputTemplate.Items[indexOutputTemplate].ToString() ?? throw new ArgumentNullException();

templateViewModel.FileName = nameSelectedTemplate;

GetTemplate?.Invoke(sender, viewTemplate);

this.commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = commandName;

commandViewModel.InputTemplate = cmdInputDoc;

commandViewModel.OutputTemplate = viewTemplate;

commandViewModel.CommandSetting = cmdSetting;

SaveCommand?.Invoke(sender, e);

List<CommandViewModel> command = new List<CommandViewModel>()

{ viewCommand };

this.SetCommandsList(command);

}

private void dataGridViewTableCommand\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

// індекс стовпця таблиці команд

string columnTableCommand = this.dataGridViewTableCommand.Columns[e.ColumnIndex].Name;

if (columnTableCommand == "CmdDeleteCommand")

{

DialogResult dialogResult = CustomMessageBox.Show("Ви впевнені, що хочете видалити команду? Видалення скасувати неможливо.",

"Видалення команди", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes) // якщо видалити команду

{

44165850.01365-01 12 01

52

if (this.dataGridViewTableCommand.SelectedRows.Count == 0) //

{

CustomMessageBox.Show("Для видалення виберіть команду зі списку.", "Видалення команди", MessageBoxButtons.OK);

}

else

{ // видалення

int index = this.dataGridViewTableCommand.CurrentCell.RowIndex;

viewNameCommand = this.dataGridViewTableCommand.Rows[index].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = viewNameCommand;

GetCommand?.Invoke(sender, viewCommand);

DeleteCommand?.Invoke(sender, e);

// видалення з таблиці

this.dataGridViewTableCommand.Rows.RemoveAt(index);

IndexRowCommandTable--;

// видалення зі списку команд у модулі генерації

this.ComboBoxGenCommandList.Items.RemoveAt(index);

this.ComboBoxGenCommandList.SelectedIndex = 0;

}

}

}

}

/// <summary>

/// [Модуль генерації]

/// Перемикач використання даних команди

/// </summary>

private void SwitchGenUseCommand\_CheckedChanged(object sender, EventArgs e)

{

if (this.SwitchGenUseCommand.Checked)

this.ComboBoxGenCommandList.Enabled = true;

else

this.ComboBoxGenCommandList.Enabled = false;

}

/// <summary>

/// [Модуль команд]

/// Редагування команди

/// </summary>

private void ButtonEditCommand\_Click(object sender, EventArgs e)

{

if (this.dataGridViewTableCommand.SelectedRows.Count == 0)

{

CustomMessageBox.Show("Для редагування виберіть команду з таблиці.", "Редагування команди", MessageBoxButtons.OK);

}

else

{

int indexCmd = this.dataGridViewTableCommand.CurrentCell.RowIndex; // індекс вибраної команди

44165850.01365-01 12 01

53

this.commandName = this.dataGridViewTableCommand.Rows[indexCmd].Cells[1].Value.ToString() ?? throw new ArgumentNullException();

commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = commandName;

GetCommand?.Invoke(sender, viewCommand);

List<string> namesCommand = new List<string>();

foreach (DataGridViewRow row in this.dataGridViewTableCommand.Rows)

{

string name = row.Cells[1].Value.ToString() ?? throw new ArgumentNullException();

if (viewCommand.NameCommand != name)

{

namesCommand.Add(name);

}

}

FormEditCommand formEditCommand = new FormEditCommand(viewCommand.NameCommand, viewCommand.InputTemplate.BookmarksFile,

viewCommand.CommandSetting, namesCommand);

formEditCommand.ShowDialog();

if (!formEditCommand.SavedChanges)

return;

commandViewModel = new CommandViewModel();

commandViewModel.NameCommand = formEditCommand.newNameCommand;

commandViewModel.CommandSetting = formEditCommand.newCommandSetting;

UpdateCommand?.Invoke(sender, e);

this.dataGridViewTableCommand.Rows[indexCmd].Cells[1].Value = viewCommand.NameCommand;

this.ComboBoxGenCommandList.Items[indexCmd] = viewCommand.NameCommand;

}

}

}

}