Plataforma para búsquedas avanzadas en entornos de datos desestructurados

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

ANEXO 4: DOCUMENTACIÓN TÉCNICA DE PROGRAMACIÓN

SEPTIEMBRE 2023

Autor

JUAN JOSÉ SALVO MATEOS

Tutoras

ANA DE LUIS REBOREDO

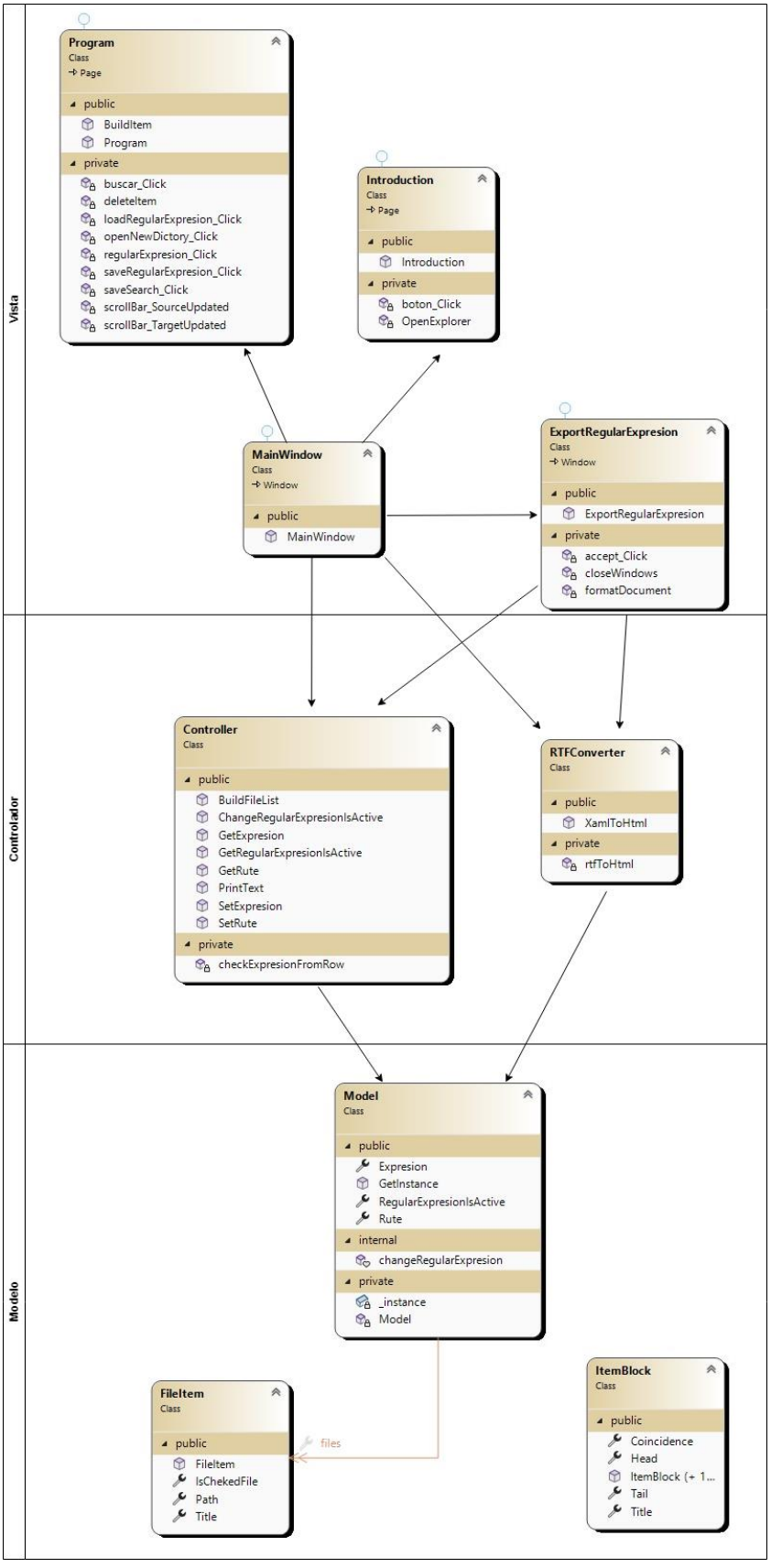
MARÍA BELÉN PÉREZ LANCHO

1. Contenido

2. Introducción

El presente documento se establece como la documentación técnica del proyecto, en el trataremos principalmente el manual para el desarrollador, documentando las clases usadas en el código y sus respectivas funciones, además se incluirá el diagrama de clases del proyecto y se adjuntará el código del proyecto.

3. Diagrama de clases del proyecto



El diagrama anterior se corresponde con el diagrama de clases utilizado en el proyecto. En él se constituyen las tres capas usadas, siguiendo el patrón arquitectónico Modelo-Vista-Controlador (MVC).

En la capa de la vista encontraremos 4 interfaces:

- *Program*: esta interfaz es donde se incluirán las funcionalidades principales de la aplicación.
- *Introduction*: En esta interfaz será la primera a la que accedamos al iniciar la aplicación, albergará la funcionalidad de carga de directorio y se podrá acceder a ella desde la interfaz *Program.*
- *MainWindow:* Esta interfaz, solamente se encargará de albergar a las otras dos interfaces anteriores, *Program* y *Introduction.*
- *ExportRegularExpresion:* Esta interfaz consiste en exportar expresiones regulares almacenadas para un futuro uso.

En la capa del controlador tendremos dos controladores que se identifican como clases estáticas, esto nos proporcionará una mayor facilidad a la hora de realizar llamadas desde las interfaces y aumentará la velocidad de la aplicación ya que no es necesario que almacenen datos debido a que hará accesos a memoria a través del modelo, el controlador principal y el controlador de generador de informes:

- *Controller*: este controlador es el controlador principal de la aplicación donde se encuentran la mayoría de las funcionalidades de la aplicación.
- *RTFConverter*: este controlador albergará las funcionalidades referentes a la generación de informes HTML.

En la capa del modelo tendremos 3 clases, una será el modelo donde se almacenarán los datos por medio del patrón singleton y dos clases que serán estructuras de datos:

- *Model*: Esta formado por un patrón singleton y albergará todos los datos no persistentes necesarios para el funcionamiento de la aplicación.
- *ItemBlock*: Esta clase servirá para tipar los datos a mostrar en el stackpanel de listado de resultados de la aplicación.

- *FileItem*: Esta clase servirá para almacenar los datos de cada archivo del directorio cargado en la aplicación.

4. Documentación de bibliotecas

Para este proyecto no se ha creado ninguna biblioteca gracias a que el por el funcionamiento del patrón Modelo-Vista-Controlador (MVC) y la albergadura del proyecto no se ha visto la necesidad de crear bibliotecas propias. Si me gustaría resaltar la biblioteca:

```
System.Text.RegularExpressions;
```

Que nos ha permitido gestionar todas las expresiones regulares que quiera introducir el usuario de forma muy sencilla ya que usa funciones muy similares a las de tipo string.

Al ser todas las bibliotecas del proyecto bibliotecas de sistema se adjuntará los enlaces de su respectiva documentación.

System: https://learn.microsoft.com/en-us/dotnet/api/system.net?view=net-7.0

System.Collections.Generic: https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic?view=net-7.0

System.IO: https://learn.microsoft.com/en-us/dotnet/api/system.io?view=net-7.0

System.Linq: https://learn.microsoft.com/en-us/dotnet/api/system.linq?view=net-7.0

System.Text.RegularExpressions: https://learn.microsoft.com/es-es/dotnet/api/system.text.regularexpressions.regex?view=net-7.0

System.Windows: https://learn.microsoft.com/en-us/dotnet/api/system.windows.documents?view=windowsdesktop-7.0

System.Windows.Controls: https://learn.microsoft.com/en-us/dotnet/api/system.windows.controls?view=windowsdesktop-7.0

Systen.Windows.Documents: https://learn.microsoft.com/en-us/dotnet/api/system.windows.documents?view=windowsdesktop-7.0

System.Windows.Markup: https://learn.microsoft.com/en-us/dotnet/api/system.windows.markup?view=net-7.0

System.Windows.Forms: https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms?view=windowsdesktop-7.0

System.Windows.Media: https://learn.microsoft.com/en-us/dotnet/api/system.windows.media?view=windowsdesktop-7.0

5. Código fuente

El código fuente será adjuntado a este proyecto, tanto el instalador en un .exe como el proyecto comprimido.

6. Manual del programador

En este apartado se documentarán las funciones de todas las capas explicado sus usos para que el programador al leer la documentación técnica este capacitado para su uso.

6.1. Class MainWindow

Lógica de intección para MainWindow.xaml

6.2. Class Program

Hereda de Page.

public Program()

Constructor de la clase.

private void Buscar_Click(object sender, RoutedEventArgs e)

Manejador para realizar la búsqueda de la expresión.

Recibe object sender y RoutedEventArgs e

private void RegularExpresion_Click(object sender, RoutedEventArgs e)

Manejador para alternar entre la funcionalidad de expresiones regulares y literales.

Recibe object sender y RoutedEventArgs e.

public Border BuildItem(ItemBlock block)

Construye el objeto para mostrar en la interfaz.

Recibe ItemBlock block.

private void DeleteItem(object sender, RoutedEventArgs e)

Manejador para eliminar un bloque de resultado de la interfaz.

Recibe object sender y RoutedEventArgs e.

private void OpenNewDictory_Click(object sender, RoutedEventArgs e)

Manejador para acceder a la vista Introduction para abrir un nuevo directorio.

Recibe object sender y RoutedEventArgs e.

private void SaveRegularExpresion_Click(object sender, RoutedEventArgs e)

Manejador para almacenar expresiones regulares.

Recibe object sender y RoutedEventArgs e.

private void LoadRegularExpresion_Click(object sender, RoutedEventArgs e)

Manejador para cargar expresiones regulares.

Recibe object sender y RoutedEventArgs e.

private void SaveSearch_Click(object sender, RoutedEventArgs e)

Manejador para generar informes.

Recibe object sender y RoutedEventArgs e.

6.3. Class Introduction

Hereda de Page. Lógica de interacción para Introduction.xaml.

public Introduction()

Inicializa la clase.

private void Boton_Click(object sender, RoutedEventArgs e)

Manejador para cuando hace el usuario hace Click en aceptar.

Recibe object sender y RoutedEventArgs e.

private void OpenExplorer(object sender, RoutedEventArgs e)

Manejador para gestionar cuando el usuario desea abrir el exporador de archivos para seleccionar un directorio.

Recibe object sender y RoutedEventArgs e.

6.4. Class ExportRegularExpresion

Lógica de intección para ExportRegularExpresion.xaml.

public ExportRegularExpresion()

Constructor que inicializa ExportRegularExpresion.

private void Accept_Click(object sender, RoutedEventArgs e)

Manejador para cuando el usuario hace click en aceptar.

Recibe object sender y RoutedEventArgs e.

private void CloseWindows(object sender, RoutedEventArgs e)

Manejador para cuando el usuario hace click en cerrar la ventana.

Recibe object sender y RoutedEventArgs e.

private string FormatDocument()

Formatea el documento a exportar

6.5. Class Controller

Controlador principal de la aplicación

public static void SetRute(string rute)

Establece la ruta del directorio en el modelo. Recibe string.

public static void SetExpresion(string expresion)

Establece la expresión usada en el modelo. Recibe string.

public static string GetRute()

Obtiene la ruta del directorio. Retorna string.

public static string GetExpresion()

Obtiene la expresión almacenada en el modelo. Retorna string.

public static bool GetRegularExpresionIsActive()

Obtiene si las expresiones regulares están activas o no. Retorna bool.

public static void ChangeRegularExpresionIsActive()

Alterna de modo entre expresiones regulares o literales en el modelo.

public static List<ItemBlock> PrintText()

Realiza las búsquedas y retorna resultados. Retorna lista de ItemBlock.

public static List<FileItem> BuildFileList()

Instancia la lista de archivos en el directorio establecido. Retorna lista de FileItem.

private static bool CheckExpresionFromRow(string row)

Comprueba si se cumple la expresión, ya sea en modo expresiones regulares o expresiones literales. Retorna bool, true si lo encontró, false en cualquier otro caso.

6.6. Class RTFConverter

Esta clase constituye el controlador para generar informes.

private static string RtfToHtml(FlowDocument flowDocument)

Transforma un texto en formato RTF a HTML. Retorna string y recibe FlowDocument.

public static string XamlToHtml(StackPanel myStackPanel)

Transforma un archivo en XAML en HTML. Retorna string y recibe un StackPanel.

6.7. Class Model

variables:

- _instance, Model, almacena la instancia singleton.
- rute, string, almacena la ruta del directorio.
- expresion, string, almacena la expresion.

- regularExpresionIsActive, bool, identifica el modo en el que se encuentra la expresión, true expresion regular, false expresion literal.
- files, lista de FileItem, almacena la lista de ficheros cargados.

public static Model GetInstance()

Método para obtener la instancia singleton.

internal void changeRegularExpresion()

Método para alternar en el modo de la expresión regular.

private Model()

Constructor del modelo.

6.8. class FileItem

Variables

- Title, string, titulo del documento.
- Path, string, ruta del documento.
- IsChekedFile, bool, identifica si el documento esta activo, true, o no, false.

public FileItem(string title, string path, bool isChekedFile)

Constructor de FileItem, recibe string title, string path, bool isChekedFile.

6.9. class ItemBlock

Clase para formatear los datos de los resultados de las búsquedas.

Variables

Title, string, título.

Head, Run, texto de línea anterior y línea actual.

Coincidence, Bold, texto que coincide con la expresión.

Tail, Run, texto de línea actual y siguiente.

public ItemBlock(string title, string head, string coincidence, string tail)

Constructor para elementos completos, recibe string title, string head y string tail.

public ItemBlock()

Constructor para elementos vacíos.