Plataforma para búsquedas avanzadas en entornos de datos desestructurados

TRABAJO DE FIN DE GRADO

GRADO EN INGENIERÍA INFORMÁTICA

SEPTIEMBRE 2023

Autor

JUAN JOSÉ SALVO MATEOS

Tutoras

ANA DE LUIS REBOREDO

MARÍA BELÉN PÉREZ LANCHO

Certificado del/los tutor/es TFG

D./Dña. Ana de Luís Rebodero, profesor/a del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Salamanca,

D./Dña. María Belén Pérez Lancho, profesor/a del Departamento de Ingeniería de Sistemas y Automática de la Universidad de Salamanca,

HACEN CONSTAR:

Que el trabajo titulado "Plataforma para búsquedas avanzadas en entornos de datos desestructurados", que se presenta, ha sido realizado por Juan José Salvo Mateos, con DNI 45132502Q y
constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado en Ingeniería Informática en esta Universidad.

Salamanca, 4 de septiembre de 2023

Fdo.:

Juan José Salvo Mateos

Resumen

El Trabajo de Fin de Grado (TFG) trata sobre el desarrollo de un programa de búsqueda de información en sistemas de archivos desestructurados. El objetivo principal es ofrecer una solución completa y eficiente que permita a los usuarios realizar búsquedas avanzadas utilizando expresiones regulares, almacenar y cargar estas expresiones, y generar informes detallados sobre los resultados de la búsqueda.

El problema que se aborda es la dificultad de buscar y analizar datos en sistemas de archivos, donde la información no sigue una estructura definida. Se revisan programas existentes y se identifican limitaciones como la falta de opciones de búsqueda avanzada y la falta de informes detallados.

La solución propuesta incluye el desarrollo de una aplicación que permite a los usuarios realizar búsquedas avanzadas utilizando expresiones regulares, lo que mejora la precisión y la relevancia de los resultados. Además, se implementa la funcionalidad de almacenar y cargar expresiones regulares para facilitar la reutilización de patrones de búsqueda frecuentes. También se generan informes detallados que proporcionan una visión general de los archivos encontrados y muestran un contexto relevante de la coincidencia de la búsqueda.

En el contexto de asesorías legales, donde cada detalle puede tener implicaciones significativas, la capacidad de generar informes detallados y contextualizados es de suma importancia. Estos informes no solo brindan una visión general de los archivos encontrados, sino que también ofrecen un contexto relevante sobre las coincidencias de búsqueda, lo que resulta esencial para el análisis y la toma de decisiones informadas.

La implementación de la solución se realiza utilizando tecnologías como *Windows Presentation Foundation* (WPF) y el *framework* .NET. Estas tecnologías permiten desarrollar una interfaz de usuario intuitiva y atractiva, y aprovechan las capacidades del entorno de desarrollo .NET para mejorar la eficiencia y el rendimiento del programa.

En resumen, el TFG se centra en desarrollar un programa de búsqueda de información en sistemas de archivos desestructurados con capacidades avanzadas de búsqueda, almacenamiento y generación de informes detallados. La solución propuesta

Comentado [BPL1]: Las palabras inglesas que uses en el texto porque habitualmente no se traducen van en cursiva

busca mejorar la precisión y la eficiencia en la búsqueda y análisis de datos en este tipo de sistemas, brindando a los usuarios una herramienta completa y fácil de usar.

Palabras Clave: Búsqueda avanzada, Expresiones regulares, Windows Presentation Foundation (WPF), Framework .NET, Sistemas de archivos desestructurados

Abstract

The Final Degree Project (TFG) is about the development of a search program in unstructured file systems. The main objective is to provide a comprehensive and efficient solution that enables users to perform advanced searches using regular expressions, store and load these expressions, and generate detailed reports on the search results.

The problem being addressed is the difficulty of searching and analyzing data in unstructured file systems, where information does not follow a defined structure. Existing programs are reviewed, and limitations are identified, such as the lack of advanced search options and detailed reports.

The proposed solution includes the development of an application that allows users to perform advanced searches using regular expressions, enhancing the precision and relevance of results. Additionally, the functionality to store and load regular expressions is implemented to facilitate the reuse of common search patterns. Detailed reports are also generated, offering an overview of the found files and presenting a relevant context of search matches.

In the context of legal consulting, where every detail can carry significant implications, the ability to generate detailed and contextualized reports is of utmost importance. These reports not only provide a general view of the located files, but also offer pertinent context regarding search matches, which is essential for analysis and informed decision-making.

The implementation of the solution is carried out using technologies such as Windows Presentation Foundation (WPF) and the .NET framework. These technologies allow the development of an intuitive and appealing user interface, leveraging the capabilities of the .NET development environment to enhance program efficiency and performance.

In summary, the TFG focuses on developing a search program in unstructured file systems with advanced search, storage, and detailed report generation capabilities. The proposed solution aims to improve precision and efficiency in searching and data analysis within such systems, providing users with a comprehensive and user-friendly tool.

Keywords: Advanced search, Regular expressions, Windows Presentation Foundation (WPF), .NET Framework, Unstructured file systems

Contenido

Tabla de Ilustraciones

1. Introducción

El siguiente documento constituye la memoria del trabajo de fin de grado del grado en Ingeniería Informática de la Universidad de Salamanca.

Este proyecto surge como solución al problema de localizar información dentro de un conjunto de archivos de texto que pueden tener cualquier grado de organización en carpetas y subcarpetas.

Además, se plantea un enfoque específico hacia el ámbito de las asesorías legales. En el contexto legal, la búsqueda y el análisis de información precisa y pertinente es fundamental para la toma de decisiones informadas y la construcción de argumentos sólidos.

Finalmente, he de destacar que en muchas ocasiones la utilización de expresiones literales no llega a satisfacer todas nuestras necesidades en la búsqueda de estos conjuntos de archivos de texto. Por ejemplo, cuando se requiere realizar búsquedas en las que queremos identificar cadenas de caracteres que se ajustan a un formato concreto, como es el caso de los números de teléfono o de DNI, es fácil identificar el patrón y obtener una expresión regular que filtre y descarte otro tipo de cadenas numéricas. En estos casos, las expresiones regulares nos ofrecerán mejores resultados y una mayor satisfacción al usuario, gracias a la capacidad de optimizar el tiempo y enfocar la información localizada de manera más precisa. Otro ejemplo de la ventaja que ofrecen las búsquedas avanzadas podemos encontrarlo si trabajamos con ficheros obtenidos mediante reconocimiento óptico de textos en papel, en los que con frecuencia aparecen errores ortográficos. Si estos errores pueden identificarse la búsqueda puede adaptarse a ellos gracias a la flexibilidad que proporcionan las expresiones regulares.

1.1. Estructura de la Memoria

Esta memoria se constituye por los siguientes apartados:

Resumen

Se presenta un resumen conciso del trabajo, abordando el problema, la solución propuesta y los resultados clave obtenidos durante el desarrollo.

Palabras Clave

Se identifican palabras clave relacionadas con el tema del trabajo, lo que facilita la búsqueda y la indexación del contenido.

Conceptos Teóricos

Se exploran los fundamentos teóricos esenciales para comprender el contexto del trabajo, como archivos desestructurados, archivos estructurados, expresiones regulares y algoritmos de búsqueda.

Objetivos del Proyecto

Se detallan los objetivos generales del proyecto, incluyendo los funcionales, técnicos y personales que guían el desarrollo y la investigación.

Técnicas y Herramientas

Se describen las técnicas y herramientas utilizadas en el desarrollo, como los lenguajes de programación (C#, XAML, HTML), el *framework* .NET y aplicaciones de diseño como Figma.

Trabajos relacionados

Se revisan trabajos previos relacionados con el tema, destacando soluciones similares, limitaciones y avances en la búsqueda en sistemas de archivos desestructurados.

Aspectos relevantes durante el desarrollo

Se enfatizan los aspectos clave del desarrollo, incluyendo el diseño de interfaz, algoritmos de búsqueda y la implementación de funcionalidades de expresiones regulares.

Funcionalidades de la aplicación desarrollada

Se presenta la solución desarrollada en detalle, abordando la búsqueda de expresiones literales y regulares en archivos TXT, almacenamiento de patrones de búsqueda y la generación de informes.

Líneas de Trabajo Futuras

Se identifican áreas en las que el proyecto podría expandirse o mejorarse en el futuro, como la búsqueda en otros tipos de archivos y la conectividad con bases de datos en red.

Conclusiones

Se resumen los logros y resultados obtenidos a lo largo del proyecto, junto con una reflexión sobre la contribución del trabajo y las lecciones aprendidas.

Referencias y bibliografía

Se presenta una lista de todas las fuentes consultadas y citadas en el trabajo, siguiendo las pautas de citación adecuadas.

2. Conceptos Teóricos

2.1. Archivos desestructurados o no estructurados

Según nos cuentan en el artículo de la página NetApp *(What Is Unstructured Data? Structured Data Vs Unstructured | NetApp, s. f.),* no estructurado significa que nuestro conjunto de datos u archivos no están predefinidos por un modelo de datos. Estos archivos pueden ser generados por el ser humano o por la máquina.

2.2. Archivos estructurados

Es la información que se suele encontrar en la mayoría de las bases de datos, son archivos de texto que se suelen mostrar con columnas con títulos y filas. Estos datos pueden ser ordenados y procesados fácilmente.

2.3. Expresiones regulares

Según el artículo en la web de Microsoft *(Regular Expression Language - Quick Reference | Microsoft Learn, 2023).*

Las expresiones regulares proveen un enfoque altamente efectivo y versátil para el procesamiento de texto. A través de una extensa notación de coincidencia de patrones, las expresiones regulares agilizan la capacidad de analizar extensos fragmentos de texto para diversos propósitos, tales como:

La búsqueda de patrones específicos de caracteres.

La validación de texto para asegurar su conformidad con un patrón predefinido, como podría ser una dirección de correo electrónico.

La extracción, edición, reemplazo o eliminación de subcadenas de texto.

La acumulación de cadenas extraídas en una colección con el fin de generar informes.

Para muchas aplicaciones que involucran el uso de cadenas o el análisis de grandes bloques de texto, las expresiones regulares se establecen como una herramienta esencial. Su capacidad para operar de manera ágil y precisa en la manipulación y extracción de información de texto las convierte en un recurso valioso y de amplia aplicación.

2.4. Expresiones literales

Son literales todas aquellas cadenas de caracteres fijas, carentes de los patrones y la flexibilidad característicos de las expresiones regulares.

2.5. Historia de usuario

En el artículo de Max Rehkopf en la web de la empresa ATLASSIAN nos resume brevemente que las historias de usuario son:

> *"Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final. Su propósito es articular cómo proporcionará una función de software valor al cliente."* (Rehkopf, s. f.)

Estas historias de usuario nos proporcionan una visión objetiva sobre los objetivos que desea nuestro usuario final y que realmente le proporcionen valor a la aplicación, estas historias se componen por:

Título: Funcionalidad a tratar.

Descripción: Breve descripción desde el punto de vista del usuario sobre lo que desea.

Criterios de acceptación: Normas a complir para valirdar la historia de usuario.

Definición de terminado: Condición para dar una historía de usuario por cerrada.

2.6. Patrones de diseño

Un patrón de diseño es una solución probada y reutilizable para un problema común en el desarrollo de software. Proporciona una estructura y un enfoque para abordar situaciones específicas de manera eficiente, basándose en la experiencia previa de otros desarrolladores. Estos patrones ayudan a simplificar el diseño y a mejorar la calidad del software al proporcionar pautas claras para la resolución de problemas recurrentes.

Patrón Singleton

El patrón Singleton asegura que haya solo una instancia de una clase en una aplicación y proporciona un punto de acceso global a esa instancia. Se utiliza para controlar recursos compartidos como bases de datos, configuraciones o registros de

eventos. Su implementación típica incluye una variable privada para almacenar la instancia y un método público para acceder a ella.

2.7. Patrones de arquitectura

Según el artículo Redalyc.Lenguajes de Patrones de Arquitectura de Software: Una Aproximación Al Estado del Arte (Jimenez-Torres et al., 2014) podemos sacar las siguientes conclusiones.

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor.

Patrón Layers

Este patrón se puede utilizar para estructurar programas que se pueden descomponer en grupos de subtareas, cada una de las cuales se encuentra en un nivel particular de abstracción. Cada capa proporciona servicios a la siguiente capa superior.

Las 4 capas más comúnmente encontradas de un sistema de información general son las siguientes:

- Capa de presentación (también conocida como capa UI)
- Capa de aplicación (también conocida como capa de servicio)
- Capa de lógica de negocios (también conocida como capa de dominio)

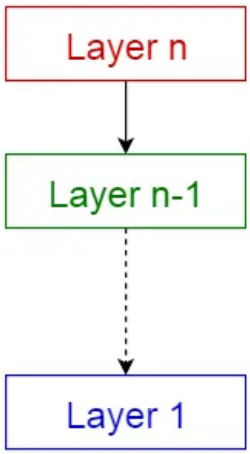

Ilustración 1 Patrón Layers

- Capa de acceso a datos (también conocida como capa de persistencia)

Patrón Modelo Vista Controlador (MVC)

El patrón de diseño Modelo-Vista-Controlador (MVC) es una arquitectura de software que se utiliza comúnmente para desarrollar aplicaciones. Su objetivo principal es organizar y separar claramente las diferentes partes de una aplicación para facilitar el desarrollo, la mantenibilidad y la escalabilidad del software.

Los componentes de definen como:

- Modelo (Model): El Modelo representa la parte central de la aplicación. Contiene la lógica de negocio y los datos de la aplicación. Es responsable de gestionar los datos y su estado, así como de realizar operaciones y cálculos relacionados con la funcionalidad de la aplicación. En resumen, el Modelo es la "mente" de la aplicación.
- Vista (View): La Vista es la parte de la aplicación que se encarga de la presentación de datos al usuario. Muestra la información de manera gráfica o textual y se comunica con el Modelo para obtener los datos que debe mostrar. La Vista no realiza ningún procesamiento de datos, simplemente muestra la información de manera legible para el usuario. Es como la "cara" de la aplicación.

- Controlador (Controller): El Controlador actúa como intermediario entre el Modelo y la Vista. Recibe las interacciones del usuario a través de la interfaz de usuario y las traduce en acciones que el Modelo debe realizar. Luego, toma los resultados del Modelo y los presenta en la Vista apropiada. El Controlador es la "mano" que ejecuta las acciones y gestiona las interacciones.

El MVC promueve la separación de preocupaciones, lo que significa que cada componente tiene una función clara y no se entremezcla con las responsabilidades de los otros componentes. Esto hace que el desarrollo sea más modular, facilita la colaboración entre equipos de desarrollo y permite realizar cambios en una parte del sistema sin afectar a las demás.

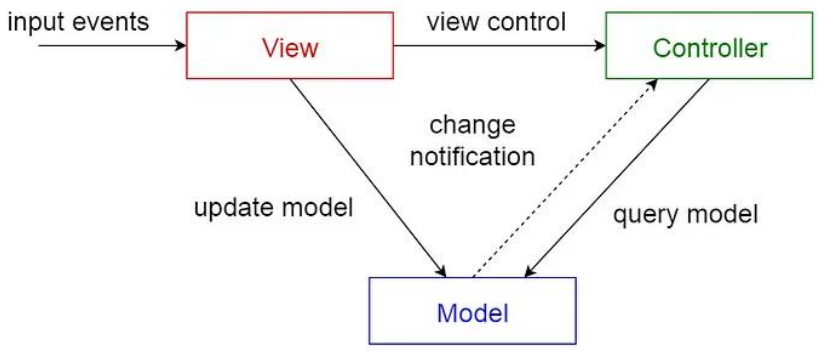

Ilustración 2 Patrón MVC

2.8. Búsqueda en profundidad

Según el artículo de López Mamani para la web Encora (López Mamani, 2020) un algoritmo de búsqueda en profundidad actúa como un algoritmo de exploración para recorrer nodos en un grafo. El funcionamiento de esta técnica radica en expandir de manera sistemática cada nodo localizado, avanzando progresivamente desde el nodo padre hacia los nodos hijos. Una vez que no quedan nodos sin visitar en una ruta específica, el algoritmo retrocede al nodo predecesor y repite el proceso con cada nodo vecino. Es importante destacar que, si se encuentra el nodo objetivo antes de recorrer todos los nodos, la búsqueda se concluye. Esta estrategia permite navegar eficientemente por la compleja estructura de datos, asegurando una exploración exhaustiva dentro del marco de este proyecto.

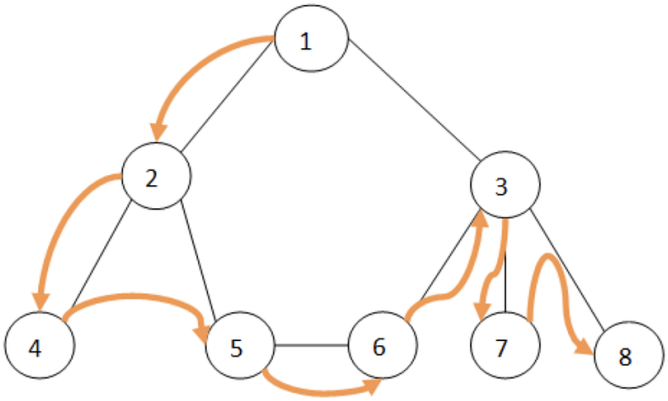

Ilustración 3 Diagrama de búsqueda en profundidad

2.9. Búsqueda en amplitud

Según el artículo de López Mamani para la web Encora (López Mamani, 2020) un algoritmo de búsqueda en amplitud opera como un algoritmo de exploración para trazar nodos en un grafo. El procedimiento de esta técnica implica un inicio en el nodo raíz (seleccionando algún nodo como nodo raíz en el caso de un grafo) y luego explorando los nodos vecinos de manera secuencial. Posteriormente, se examinan los vecinos de estos nodos adyacentes, ampliando así la exploración hasta cubrir todo el grafo. Vale la pena destacar que, si se encuentra el nodo objetivo antes de recorrer todos los nodos, la búsqueda se concluye.

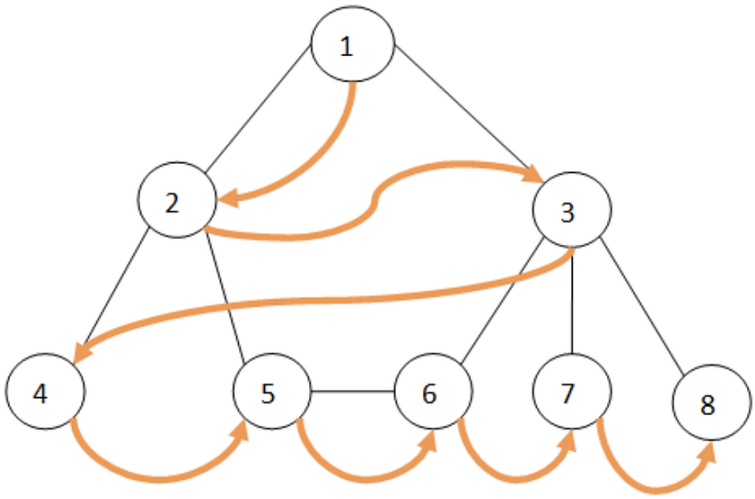

Ilustración 4 Diagrama de búsqueda en amplitud

2.10. Algoritmos de búsquedas

Según la definición que nos da Jahaziel Ponce en su artículo sobre algoritmos de búsqueda

"Un algoritmo de búsqueda es un conjunto de instrucciones que están diseñadas para localizar un elemento con ciertas propiedades dentro de una estructura de datos; por ejemplo, ubicar el registro correspondiente a cierta persona en una base de datos, o el mejor movimiento en una partida de ajedrez." (Ponce, 2021)

Por lo que podemos afirmar que los algoritmos de búsqueda nos proporcionarán una herramienta para que, teniendo una serie de datos, podamos encontrar el elemento deseado en el menor tiempo posible y de la forma más optima.

Existen tres tipos de algoritmos de búsqueda:

- **Búsqueda secuencial:** Consiste en comparar secuencialmente el elemento deseado con los elementos de una lista, comenzando siempre por el inicio de la lista hasta el final de esta. El proceso terminará cuando encontremos el elemento deseado o cuando alcancemos el final de la lista.

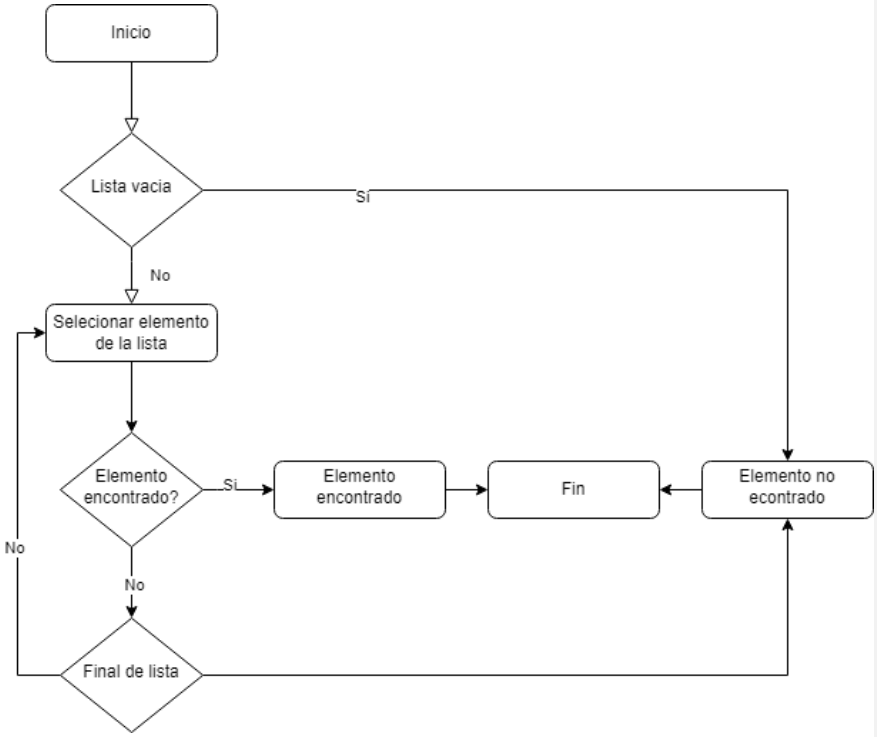

Ilustración 5 Diagrama de búsqueda secuencial.

Búsqueda secuencial indexada: Si el conjunto de datos del que disponemos es ordenable se puede realizar una búsqueda secuencial indexada. El objetivo de este algoritmo es optimizar el tiempo que tardará en ofrecernos el resultado frente al algoritmo anterior.

Búsqueda binaria: Si el conjunto de datos del que disponemos es ordenable se puede realizar una búsqueda binaria. El objetivo de este algoritmo es optimizar el tiempo que tardará en ofrecernos el resultado.

Por ello nos situaremos en medio de nuestro conjunto de datos y comprobaremos si nuestro elemento es menor, mayor o igual que el elemento seleccionado del conjunto. Si es menor nos situaremos en la mitad de conjunto de elementos menores de la lista; si es mayor, en el conjunto de elementos mayor y finalmente si es idéntico habremos finalizado. También daremos por finalizado el algoritmo si no quedan conjuntos de elementos a los que podamos comparar el elemento buscado.

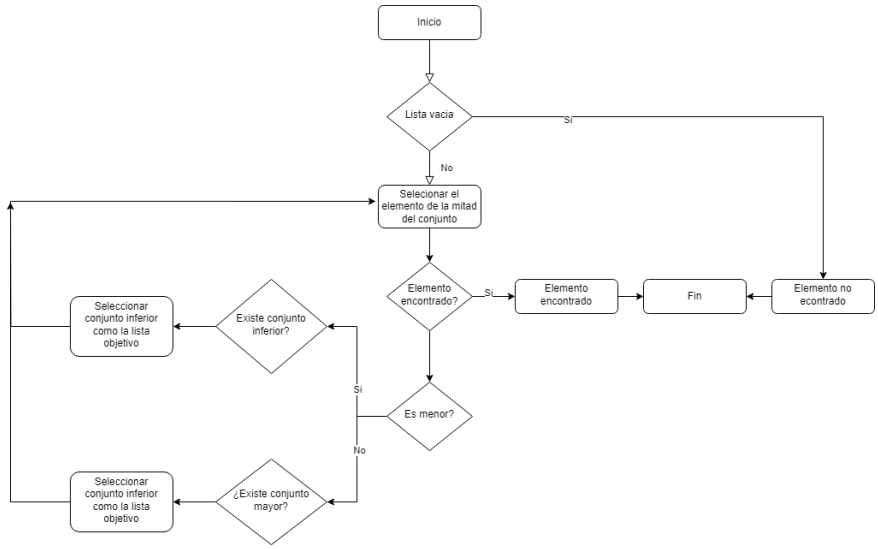

Ilustración 6 Diagrama de flujo búsqueda binaria.

3. Objetivos del proyecto

Detallaremos los objetivos principales que nos hemos planteado. Estos objetivos abarcan aspectos funcionales, técnicos y personales con el fin de guiar nuestro trabajo de investigación y desarrollo.

3.1. Objetivos funcionales

1. Desarrollar un programa de búsqueda en sistemas de archivos desestructurados que permita a los usuarios realizar búsquedas avanzadas y precisas utilizando expresiones regulares, esto les brindará a los usuarios la flexibilidad de definir patrones de búsqueda complejos, mejorando la precisión y la relevancia de los resultados obtenidos.

2. Implementar la capacidad de almacenar y cargar expresiones regulares para facilitar la reutilización y la configuración de búsquedas personalizadas. Esto permitirá ahorrar tiempo y esfuerzo. De esta manera, los usuarios podrán reutilizar patrones de búsqueda frecuentes y evitarán tener que ingresar manualmente las expresiones cada vez que realicen una búsqueda similar a otra realizada previamente.

3. Generar informes sobre los resultados de las búsquedas, incluyendo información relevante sobre los archivos encontrados y una visualización del contexto donde aparecen las coincidencias. Esta funcionalidad nos ofrecerá la capacidad de obtener y almacenar informes detalladosque faciliten el posterior análisis de los resultados de la búsqueda. Estos informes proporcionarán una visión general de los archivos encontrados, incluyendo detalles como nombre del archivo, su ubicación y la identificación del fragmento de texto donde se encuentre la coincidencia de la búsqueda. Esto permitirá a los usuarios obtener una vista previa rápida y relevante del contenido de los archivos, facilitando el análisis y la toma de decisiones.

3.2. Objetivos técnicos

1. Utilizar el lenguaje de programación C# y el *framework* .NET para desarrollar el programa de búsqueda en sistemas de archivos desestructurados.

2. Aplicar la tecnología Windows Presentation Foundation (WPF) para crear una interfaz de usuario moderna, intuitiva y atractiva. La interfaz permitirá ingresar las expresiones regulares, definir los criterios de búsqueda y realizar las consultas de manera sencilla. Además, los informes generados serán presentados de forma clara y legible, facilitando la comprensión de los resultados.
3. Implementar técnicas eficientes de manipulación de archivos y procesamiento de texto para mejorar el rendimiento y la precisión de las búsquedas.

3.3. Objetivos personales

1. Abordar todas las etapas de la definición, diseño y desarrollo de una aplicación de escritorio.
2. Ampliar conocimientos en el manejo de expresiones regulares.
3. Mejorar habilidades de diseño de interfaces de usuario y experiencia de usuario (UI/UX).

4. Técnicas y herramientas

Para llevar a cabo este proyecto, se ha decidido implementar la solución utilizando la tecnología Windows Presentation Foundation (WPF) y el *framework* .NET. Estas tecnologías proporcionan una base sólida y versátil para el desarrollo de aplicaciones de escritorio, con una interfaz de usuario moderna y atractiva.

La elección de WPF como tecnología principal para la interfaz de usuario permitirá crear una experiencia visualmente agradable y altamente interactiva para los usuarios. WPF ofrece una amplia gama de controles y herramientas de diseño que facilitan la creación de interfaces intuitivas y personalizadas. Además, al aprovechar las capacidades gráficas y multimedia de Windows, se brinda la oportunidad de mejorar la presentación de los informes generados y proporcionar una visualización atractiva del contexto de los documentos.

El uso del *framework* .NET permitirá aprovechar las ventajas de un entorno de desarrollo robusto y ampliamente utilizado. .NET ofrece una gran variedad de bibliotecas y componentes que simplificarán el desarrollo de las funcionalidades clave de la solución, como la manipulación de archivos, el manejo de expresiones regulares y la generación de informes, lo que le hace adecuado para los objetivos de nuestro proyecto. Además, al elegir .NET, se garantiza una mayor compatibilidad con diferentes versiones de Windows y una mayor facilidad para integrar tecnologías adicionales en el futuro, si fuese necesario.

La combinación de WPF y .NET brindará un entorno de desarrollo sólido y eficiente para implementar la solución propuesta. Se aprovecharán las características y capacidades de ambas tecnologías para ofrecer una aplicación de búsqueda en sistemas de archivos desestructurados altamente funcional, intuitiva y estéticamente agradable.

Con esta implementación, se espera lograr un producto final de alta calidad que cumpla con los requisitos establecidos y ofrezca una experiencia de usuario satisfactoria.

4.1. Lenguajes de programación

4.1.1. C#

Es un lenguaje de programación orientado a objetos, es decir, pertenece a un paradigma de programación que se basa en el concepto de clases y objetos. Según el

artículo *¿Qué es la Programación Orientada a Objetos? (Martínez Canelo, s. f.)*, este tipo de programación se utiliza para estructurar un programa software en piezas simples y reutilizables de planos de código (clases) para crear instancias individuales de objetos.

Las características de este lenguaje por la publicación de la web Desarrollo web *(C# Por DesarrolloWeb.Com, s. f.)* son:

- Multiplataforma: es ejecutable en los sistemas más comunes como Windows, MacOs o Linux.
- Sintaxis similar a C, C++ y Java.
- Lenguaje de paradigma de programación orientada a objetos, con expresiones de control heredadas de la programación estructurada.
- Fuertemente tipado: no se permiten violaciones de los tipos de datos, es decir, el valor de una variable de un tipo concreto no se puede usar como si fuera de un tipo distinto a menos que se haga una conversión.
- Lenguaje moderno con actualizaciones de mejora frecuentes.
- Dispone de un nutrido conjunto de librerías.
- Orientado a componentes: permite una programación con interfaces y dependencias bien definidas que permiten ofertar o solicitar un conjunto de servicios o funcionalidades.

4.1.2. XAML

Según la web de Microsoft (XAML *Overview - UWP Applications | Microsoft Learn, s. f.*) es un lenguaje declarativo para objetos y sus propiedades, pero también incluye una sintaxis para adjuntar controladores de eventos a los objetos del marcado. La sintaxis de evento de XAML puede integrar los eventos declarados mediante XAML con el modelo de programación de Windows Runtime, que según él artículo de J. Serrano *(J. Serrano, s. f.)* es un modelo de programación implementado y presentado por Microsoft que facilita el desarrollo de aplicaciones tanto para el chip Intel como para el chip ARM.

Pero lo realmente interesante es que el uso de XAML se ha ampliado para que podamos utilizarlo para crear aplicaciones WPF en Silverlight o con C++.

Las características de XAML son:

- Jerarquías de elementos bien definidas: por la forma de trabajar con XAML podemos visualizar con mayor facilidad la jerarquía de estos, y diferenciar bien entre los elementos primarios y secundarios de una interfaz de usuario.
- Suele ser más conciso y legible que el código equivalente en otros lenguajes.

4.1.3. HTML

El HTML o Lenguaje de Marcas de Hipertexto (del inglés *HyperText Markup Language)*, es un lenguaje de etiquetas, es decir, utiliza "marcas" para etiquetar el texto, imágenes y vídeos.

Según la web MDN (*HTML: Lenguaje De Etiquetas De Hipertexto | MDN*, s. f.), un elemento HTML se distingue de otro texto en un documento mediante la señalización con los signos "<", *">"* y *"/>"*. Algunos ejemplos de estos elementos son <*Head*>, <*Titile*>, <*Body*>, <*Img*> o <*Nav*>. Cada una de ellas dota de unas funcionalidades diferente para visualizar su contenido de una forma u otra dependiendo del uso que le deseemos dar.

Las características de HTML según Next_U *(Qué Es HTML: Guía Sobre Este Lenguaje De programación básico, s. f.)* son:

- Fácil de usar y entender.
- Es utilizado para crear páginas web.
- Permite describir hipertextos.
- Permite que el usuario se mueva por cualquier sitio de internet haciendo clic en un texto especifico: hipervínculos.
- Está fundamentado por una serie de breves códigos escritos en un registro de texto por el desarrollador del sitio web.
- Es multiplataforma, por lo que se puede acceder desde cualquier lugar y dispositivo.
- No es necesario estar en línea para que el lenguaje HTML funcione correctamente, ya que se puede codificar sin conexión alguna.

- Todos los elementos de un documento HTML constan de una etiqueta de inicio, un bloque de texto y una etiqueta de cierre.
- Tiene un despliegue rápido.
- Es reconocido y admitido por cualquier tipo de explorador web.
- Permite archivos pequeños.
- Su lenguaje es estático.
- Las etiquetas son limitadas.
- Ha tenido revisiones y actualizaciones a lo largo del tiempo.
- Es fácil crear archivos HTML desde otros lenguajes de programación.

4.2. Framework

Un *Framework* es un marco o esquema que agiliza nuestro trabajo programando, evitando que debamos escribir código de manera repetitiva. El uso de estos *framework* nos permitirá ahorrar tiempo a la hora de programar nuestra aplicación, nos aporta seguridad a nuestra aplicación ya que gran parte de las vulnerabilidades que pueda llegar a tener nuestro programa quedan resueltas, podremos acceder a código de la comunidad para usarlo cómodamente en nuestra aplicación si fuese necesario.

Algunos ejemplos de los *frameworks* más populares son:

- .NET
- Symphony
- Zend Framework
- Laravel
- Ruby on Rails
- Angular

4.2.1. .NET

En la web de Microsoft *(C# Docs - Get Started, Tutorials, Reference. | Microsoft Learn, s. f.),* encontramos que es un entorno que nos proporciona diversos servicios a las aplicaciones de ejecución. Está formado por dos componentes esenciales: *Common Languaje Runtine* (CLR), que es el motor de ejecución que controla las aplicaciones; y la biblioteca de clases de .NET *Framework,* que proporciona una biblioteca con una gran variedad de código probado y reutilizable al que podemos llamar desde nuestras propias aplicaciones.

Entre sus características observamos:

- Administración de la memoria: en otros lenguajes como C debemos asignar o liberar memoria en tiempo de ejecución mediante un código explícito. Este no es el caso de .NET. Este se encargará dinámicamente de manejar la memoria que ocupemos en nuestras aplicaciones.
- Sistemas de tipos comunes: en los lenguajes de programación tradicionales, el compilador define los tipos básicos, lo que nos limita a la hora de trabajar con otros lenguajes. Pero gracias a . NET, no tendremos que preocuparnos de ello ya que nos facilitará un sistema de tipos de . NET *Framework* y son comunes a todos los lenguajes que tienen como destino . NET *Framework.*
- Bibliotecas de clases extensas: para evitar que debamos desarrollar extensas cantidades de código, . NET nos ofrecerá una gran variedad de bibliotecas que tendremos a nuestra entera disposición para poder realizar nuestras tareas de programación de bajo nivel de una forma sencilla y cómoda.
- Marcos y tecnologías de desarrollo: . NET incluye bibliotecas para determinadas áreas como *ASP.NET, ADO.NET, Windows Comunication Fundation* y *Windows Presentation Foundation.*
- Interoperabilidad entre lenguajes: los compiladores de lenguajes cuya plataforma de destino es .NET Framework emiten un código intermedio denominado Lenguaje intermedio común (CIL), que, a su vez, se compila en tiempo de ejecución a través de *Common Language Runtime*. Con esta característica, las rutinas escritas en un lenguaje son accesibles para otros lenguajes, de modo que

los programadores puedan centrarse en crear aplicaciones en su lenguaje preferido.

- Compatibilidad de versiones: las aplicaciones desarrolladas en una versión .NET raramente deben ser modificadas en versiones posteriores.
- Ejecución en paralelo: ayuda a resolver conflictos entre versiones y permite que varias versiones de *Common Language Runtime* coexistan en el mismo equipo. Esto significa que pueden utilizarse varias versiones de las aplicaciones, y que una aplicación se puede ejecutar en la versión de .NET *Framework* con la que se compiló.
- Compatibilidad con múltiples versiones: al establecer .NET *Standard* como destino, los desarrolladores pueden crear bibliotecas de clases que funcionan en varias plataformas de .NET *Framework* compatibles con esa versión del estándar.

  Como nuestro proyecto esta realizado en WPF que es un framwork de .NET, estaremos muy lígados a este y nos proporcionará grandes facilidades a la hora de implementar algoritmos de búsqueda.

4.2.2. WPF

En la documentación de Microsoft *(Getting Started - WPF .NET Framework | Microsoft Learn, s. f.),* se define que *Windows Presentation* Foundation (WPF), además de un subconjunto de .NET, es un marco para desarrollar interfaces de usuario. Admite un amplio conjunto de características de desarrollo de aplicaciones, incluidos un modelo, controles, gráficos, diseños, enlaces a datos, entre otros. WPF utiliza el lenguaje XAML para proporcionar un modelo declarativo para la programación de aplicaciones.

4.3. Aplicaciones de diseño

4.3.1. Figma

Según el artículo de Jeffrey Paredes para la página de la escuela británica de artes creativas y tecnología *(Paredes, 2023).* Es un editor gráfico parecido a Sketch y Adobe XD utilizado por diseñadores y programadores, que se utiliza para diseñar interfaces gráficas para posteriormente el programador pueda traducir los diseños a código para facilitar su programación de una forma rápida y eficiente.

Figma utiliza un sistema de gráficos vectoriales, es decir, según el artículo de la Universidad Nacional de Quilmes *(Gráficos Vectoriales, s. f.),* funciona mediante un sistema de imágenes generadas con segmentos de línea conectados por nodos y resultan de un promedio sobre las dos tangentes creados por los nodos o puntos de control, estos vectores pueden ser curvos o rectos según determinen los manejadores que parten de los nodos.

Finalmente resaltar brevemente que Figma nos proporciona una gran variedad de herramientas que nos favorecerán a la hora de realizar diseños como *grids* y *layouts*; capas y grupos; estilos; componentes; *auto layouts* y *plugins.*

5. Trabajos relacionados

En el campo de la Ingeniería Informática, existen numerosos programas que permiten realizar búsquedas en sistemas de archivos estructurados. Sin embargo, en el caso de los sistemas de archivos desestructurados, donde los datos no siguen una organización rígida, la búsqueda se vuelve más compleja. Actualmente, no existen muchas soluciones específicas que aborden eficientemente esta problemática.

Ejemplos de programas:

1. ***Google Desktop Search*:** Es un programa de búsqueda de archivos para sistemas operativos Windows. Permite realizar búsquedas en el sistema de archivos local, incluyendo documentos, correos electrónicos, chats, archivos multimedia, entre otros. Sin embargo, se centra en sistemas de archivos estructurados y no ofrece la capacidad de utilizar expresiones regulares.
2. ***Copernic Desktop Search*:** Similar a Google Desktop Search, Copernic Desktop Search es una herramienta de búsqueda de archivos para Windows. Permite buscar archivos en el sistema local, incluyendo correos electrónicos, documentos, archivos multimedia y más. Aunque ofrece algunas opciones de búsqueda avanzada, no proporciona una funcionalidad específica para sistemas de archivos desestructurados o el uso de expresiones regulares.
3. ***dtSearch*:** Es una herramienta de búsqueda de texto completa que permite a los usuarios buscar en una amplia variedad de archivos y formatos, incluyendo documentos, correos electrónicos, bases de datos, archivos PDF y más. Ofrece opciones de búsqueda avanzadas y soporte para expresiones regulares. Sin embargo, no se centra en sistemas de archivos desestructurados y no proporciona funcionalidades específicas para almacenar y cargar expresiones regulares ni generar informes detallados sobre los resultados de la búsqueda.

5.1. Problemas

Los programas existentes, como *Google Desktop Search, Copernic Desktop Search y dtSearch*, han demostrado ser herramientas útiles para la búsqueda de archivos en sistemas operativos y entornos de escritorio. Sin embargo, estos programas presentan limitaciones significativas cuando se trata de lidiar con sistemas de archivos

Comentado [AdLR2]: También tienes pendiente explicar con más claridad a qué te refieres con archivos estructurados, tal y como te comenté en la anterior versión.

Comentado [JS3R2]: Se introduce en el punto de conceptos teóricos

desestructurados y utilizar expresiones regulares. Estos problemas pueden dificultar la búsqueda precisa y eficiente, así como el análisis de los datos recuperados.

En primer lugar, la mayoría de estos programas están diseñados principalmente para buscar archivos en sistemas de archivos estructurados. Esto significa que no son capaces de abordar de manera efectiva la búsqueda en sistemas de archivos desestructurados, donde la organización y la estructura de los datos no siguen patrones predefinidos. Como resultado, los usuarios se encuentran limitados en su capacidad para encontrar y recuperar información específica en estos sistemas.

En segundo lugar, estos programas carecen de la capacidad de utilizar expresiones regulares, que son una herramienta poderosa para realizar búsquedas avanzadas y precisas. Las expresiones regulares permiten a los usuarios definir patrones de búsqueda complejos y flexibles, lo que les permite refinar aún más los criterios de búsqueda. La falta de soporte para expresiones regulares en los programas existentes limita la precisión de las búsquedas y puede llevar a resultados irrelevantes o incompletos.

Otro problema común es la falta de funcionalidad para almacenar y cargar expresiones regulares. Los usuarios de los programas existentes a menudo se ven obligados a ingresar manualmente las expresiones regulares cada vez que realizan una búsqueda, lo cual es tedioso y propenso a errores. La capacidad de almacenar y cargar expresiones regulares permitiría a los usuarios reutilizar fácilmente patrones de búsqueda frecuentes y ahorrar tiempo y esfuerzo en configurar cada búsqueda.

Por último, los programas existentes suelen carecer de la capacidad de generar informes detallados sobre los resultados de la búsqueda. La generación de informes es una funcionalidad importante para analizar y comprender los datos recuperados. Sin ella, los usuarios pueden tener dificultades para extraer información significativa de los resultados de búsqueda y realizar un análisis adecuado de los mismos.

6. Aspectos relevantes del desarrollo

6.1. Análisis

En la etapa de análisis de esta aplicación me centre en la búsqueda de requisitos, en generar historias de usuario con dichos requisitos, en elegir un buen patrón de arquitectura en el que basar nuestra aplicación para facilitar el desarrollo y desarrollar un diagrama de clases para su implementación.

6.1.1. Búsqueda de requisitos:

Para esta búsqueda de requisitos me puse en contacto son Sara Samper Henao, gerente de la Asesoría Famsam en Madrid, Torrejón. En la charla con ella me comento que este tipo de aplicaciones son muy útiles porque les permite tener un mejor control de los clientes, periodos de los impuestos de todos ellos y su utilidad tanto para su departamento económico cómo legal. Para ella uno de los requisitos más imprescindibles que comento fueron los siguientes:

1. Poder realizar búsquedas en sistemas de archivos desestructurados.
2. Poder acceder a una función de búsquedas avanzadas.
3. Poder generar informes sobre las búsquedas.
4. Poder utilizar la aplicación de forma intuitiva y simple.

6.1.2. Historias de usuario

Las historias de usuario sirven para poder definir los objetivos a conseguir de un requisitos o funcionalidades, nos proporcionan un enfoque más realista de lo que desea el usuario, nos facilita priorizar unos objetivos sobre otros al programar, nos aportan una flexibilidad ya que se pueden ajustar a medida que se avanza el desarrollo, nos permite tener una mayor comprensión de que criterios debemos cumplir para dar una historia de usuario como finalizada,

A continuación, desarrollo las historias de usuario relacionadas con los requisitos previamente presentados:

Búsqueda Eficiente en Sistemas de Archivos Desestructurados

Título: Búsqueda Eficiente en Sistemas de Archivos Desestructurados

Descripción:

Como usuario interesado en gestionar eficientemente mi información en sistemas de archivos desestructurados, deseo poder realizar búsquedas precisas y rápidas dentro de estos sistemas. Esto me permitirá encontrar de manera efectiva documentos e información relevante, incluso cuando los archivos no sigan una estructura organizativa definida.

Criterios de Aceptación:

- La búsqueda debe ser capaz de rastrear archivos y carpetas en todos los niveles del sistema, garantizando que ningún archivo relevante quede sin ser considerado.
- El sistema debe proporcionar resultados precisos y relevantes basados en las palabras clave o expresiones regulares ingresadas.
- Se deben mostrar resultados detallados que incluyan información sobre la ubicación de los archivos encontrados y un contexto relevante de la coincidencia.
- La búsqueda debe ser eficiente y rápida, garantizando tiempos de respuesta adecuados incluso para grandes volúmenes de datos.

Definición de Terminado:

La funcionalidad de búsqueda en sistemas de archivos desestructurados está implementada y ha sido probada exhaustivamente para asegurar su eficiencia y precisión. Los resultados se presentan de manera comprensible y se cumplen todos los criterios de aceptación definidos por el usuario.

Acceso a Búsquedas Avanzadas con Expresiones Regulares

Decidí centrar este requisito en expresiones regulares porque es una herramienta muy potente que puede ofrecer al usuario una gran versatilidad y eficiencia al realizar búsquedas.

Título: Acceso a Búsquedas Avanzadas con Expresiones Regulares

Descripción:

Como usuario que necesita realizar búsquedas más específicas y avanzadas en sistemas de archivos desestructurados, deseo acceder a una función de búsquedas avanzadas basada en expresiones regulares dentro de la aplicación. Esto me permitirá refinar mis consultas y obtener resultados más precisos y relevantes, especialmente en situaciones en las que necesito encontrar información altamente específica utilizando patrones de búsqueda avanzados.

Criterios de Aceptación:

- Como usuario, quiero tener acceso a una función de búsquedas avanzadas basada en expresiones regulares claramente identificada en la interfaz de usuario.
- La función de búsquedas avanzadas basada en expresiones regulares debe permitirme definir patrones de búsqueda complejos utilizando expresiones regulares estándar.
- La función de búsquedas avanzadas basada en expresiones regulares debe proporcionar resultados precisos y relevantes, presentándolos de manera comprensible y fácil de interpretar.

Definición de Terminado:

La función de búsquedas avanzadas basada en expresiones regulares está implementada y accesible desde la interfaz de usuario. Permite a los usuarios utilizar expresiones regulares. Los resultados de las búsquedas avanzadas cumplen con los criterios de aceptación definidos y se presentan de manera clara y comprensible para el usuario.

Generación de Informes sobre Búsquedas

Título: Generación de Informes sobre Búsquedas

Descripción:

Como usuario que necesita llevar un registro y documentación de las búsquedas realizadas en sistemas de archivos desestructurados, deseo poder generar informes detallados sobre los resultados de mis búsquedas. Esto me permitirá tener un registro organizado y completo de los archivos encontrados, así como compartir fácilmente estos resultados con otros interesados.

Criterios de Aceptación:

- Como usuario, quiero tener acceso a una función de generación de informes claramente identificada en la interfaz de usuario.
- La función de generación de informes debe permitirme seleccionar los resultados específicos que deseo incluir en el informe.
- Los informes generados deben estar formateados de manera legible y organizada, incluyendo detalles sobre la ubicación de los archivos encontrados y un contexto relevante de la coincidencia.

Definición de Terminado:

La función de generación de informes sobre búsquedas está implementada y accesible desde la interfaz de usuario. Los informes generados cumplen con los criterios de aceptación definidos y se presentan de manera clara y organizada para el usuario.

Experiencia de Usuario Intuitiva

Título: Experiencia de Usuario Intuitiva

Descripción:

Como usuario de la aplicación, deseo poder utilizarla de manera intuitiva y sencilla para que mi experiencia sea lo más fluida y sin complicaciones posible. Esto es crucial para que pueda aprovechar todas las funcionalidades de la aplicación sin enfrentar obstáculos innecesarios.

Criterios de Aceptación:

- La interfaz de usuario debe ser limpia y organizada, con una disposición lógica de elementos y opciones claramente etiquetadas.
- La navegación dentro de la aplicación debe ser intuitiva, permitiendo a los usuarios moverse fácilmente entre las diferentes funcionalidades y secciones.
- Se debe minimizar la necesidad de capacitación o documentación externa para que los usuarios puedan comenzar a utilizar la aplicación de inmediato.

- La aplicación debe ser consistente en su diseño y funcionamiento, siguiendo estándares de usabilidad comunes.

Definición de Terminado:

Los usuarios pueden utilizar la aplicación de forma intuitiva y sencilla sin la necesidad de una curva de aprendizaje significativa. La interfaz de usuario es organizada y clara, y se proporcionan instrucciones y ayuda cuando sea necesario para garantizar una experiencia de usuario óptima. La navegación dentro de la aplicación es intuitiva y se siguen principios de diseño de usabilidad para garantizar la consistencia en toda la aplicación.

6.1.3. Patrón arquitectónico

Para el patrón arquitectónico me planteé varias opciones, entre ellos el patrón *Layers* y el patrón Modelo-Vista-Controlador (MVC). Finalmente me decidí aplicar el patrón Modelo-Vista-Controlador, debido a que a diferencia del patrón *Layers*, El patrón MVC nos permite un mejor manejo al adaptarse mejor a nuestra aplicación, debido a que la capa de la Vista se quedará definida como nuestros archivos XAML y sus archivos C#; En la parte del controlador implemente las funcionalidades, además dependiendo de las acciones que deseemos hacer podremos acceder a un controlador u a otro. Finalmente, en el modelo, almacenaremos los datos para que sean accesibles en todos los controladores.

6.1.4. Almacenamiento de datos

Para que el almacenamiento de datos sea accesible desde cualquier controlador, necesite implantar el patrón Singleton. Este patrón nos ofrece la posibilidad de crear una única instancia accesible desde cualquier controlador para obtener un único punto de acceso a la base de datos, así podremos acceder a cualquier dato que necesitemos del modelo, refinarlos y devolvérselos a la vista para que los muestre desde el controlador correspondiente.

6.1.5. Diagrama de clases

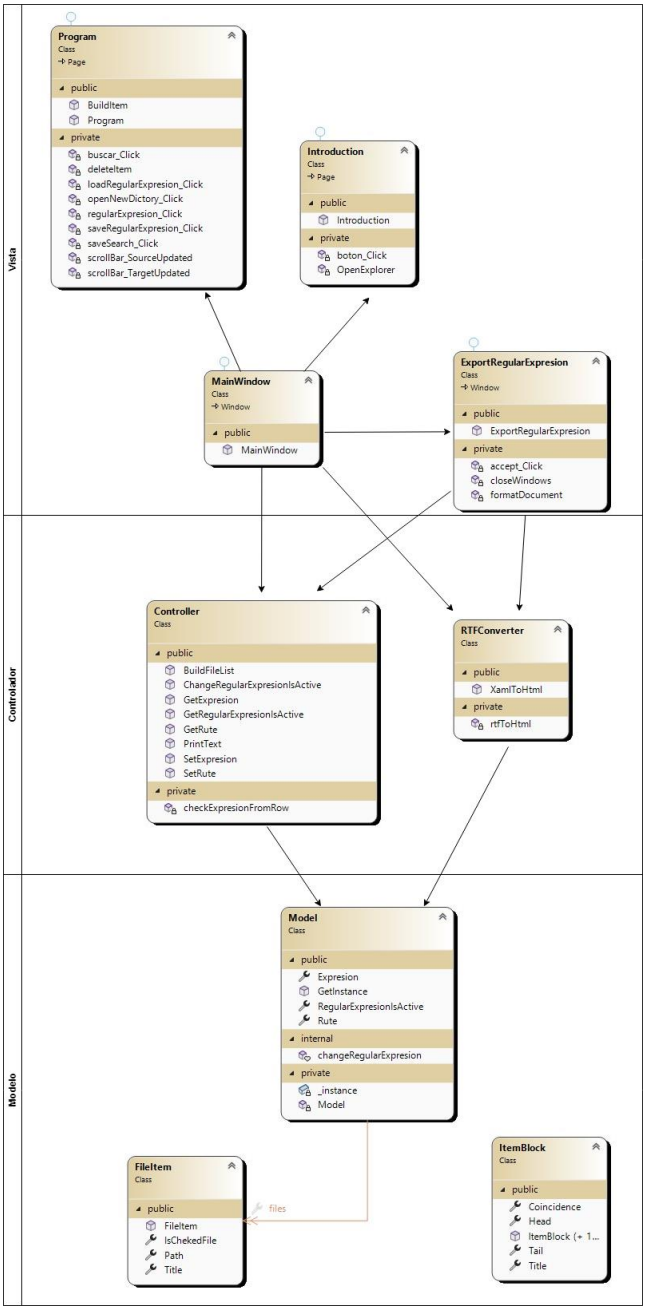

Ilustración 7 Diagrama de clases

En el presente diagrama de clases podremos obtener una visión global de la aplicación nos proporciona una idea de los métodos implementados y de las variables usadas.

6.2. Diseño de interfaz

Para el diseño de la aplicación comencé creando una guía de estilos, según el artículo de la página 3Androides *(¿Por Qué Debes Tener Una guía De Estilo Antes De Comenzar Tu App, s. f.)* Estas guías son manuales que recogerán el aspecto visual de todos los elementos que se utilizarán durante el desarrollo de nuestra aplicación, así a la hora de programar los elementos visuales ya tendremos una idea previa de tipografías, tamaños de letra, maquetación y paleta de colores a utilizar.

Para facilitar este trabajo utilicé la aplicación web Figma. Figma es una herramienta de prototipado web y editor de gráficos vectorial, que como he mencionado con anterioridad al ser una aplicación web, no necesita ninguna instalación para su uso. Esta aplicación nos proporciona una interfaz intuitiva y muchísimas opciones para elaborar diseños, lo más destacable de ella es que contiene todos los estilos disponibles de WPF y además gracias a que nos muestra el código en CSS, un lenguaje utilizado para aplicar estilos en HTML, podremos traducirlo con muy poco esfuerzo a nuestros estilos de XAML.

Para la guía de diseño decidí seleccionar la tipografía Calibri, que según el artículo de Graffica *(¿Quién diseñó Calibri? La icónica tipografía De Microsoft)* es una tipografía diseñada por Lucas de Groot especialmente para Microsoft. Esta tipografía dispone de unos trazos muy cuidados, unas esquinas ligeramente redondeadas que trasmiten una sensación más amistosa que otras tipografías y unas proporciones humanísticas, que se diseñaron específicamente para su uso en pantallas, sin embargo, es una tipografía que muy legible en papel impreso. La prueba de la relevancia de esta tipografía es que Calibri fue galardonada en la competición de diseño tipográfico en 2005 bajo el título de “Tipografías de sistema” celebrado por el Club de Directores Tipográficos.

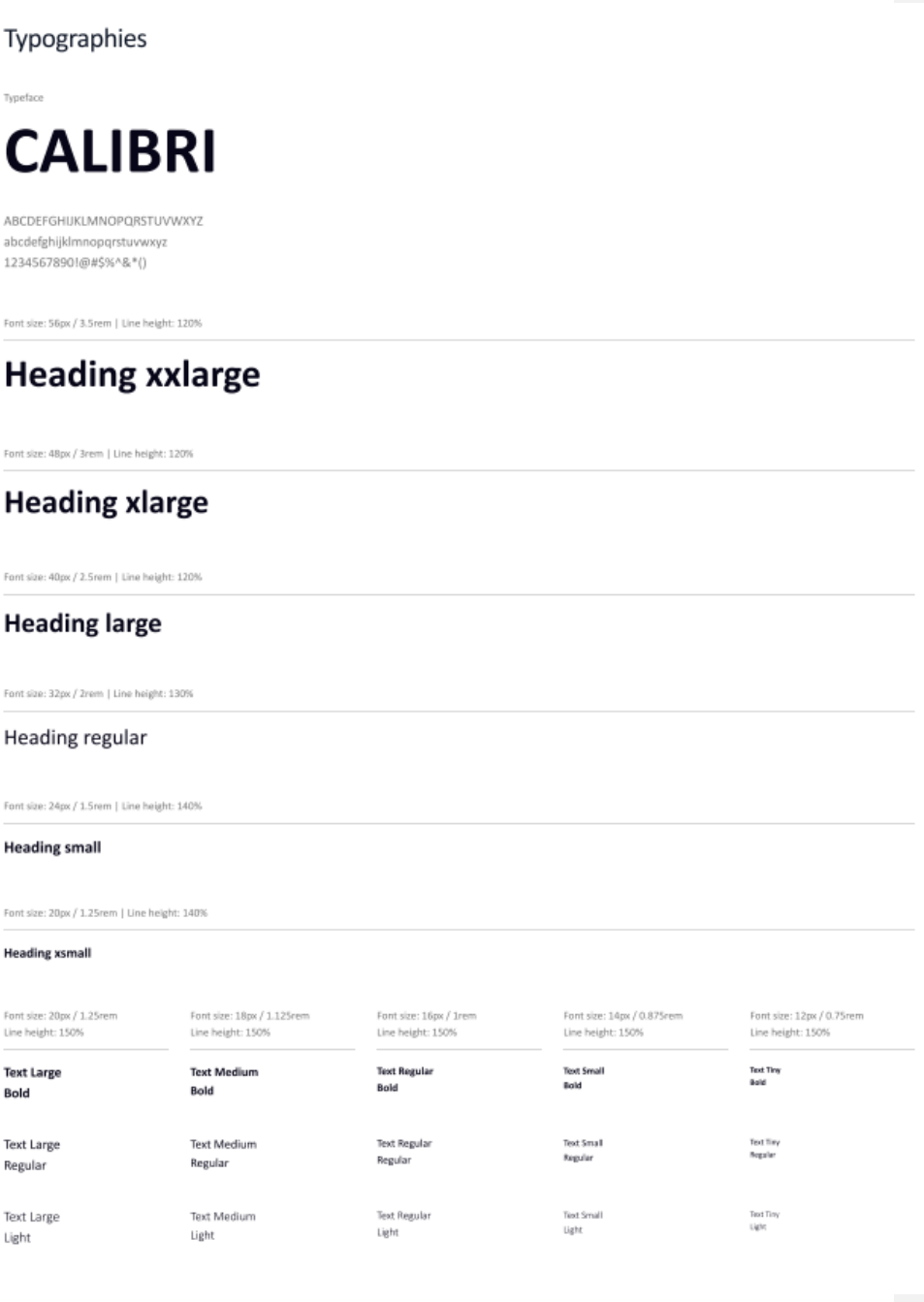

Ilustración 8 Guía de estilos de tipografías.

Para la paleta de colores seleccioné los siguientes:

- **Gris medio (#999999):** El gris medio es un color neutral y equilibrado que puede transmitir una sensación de estabilidad y confianza. Es un color ideal para ser utilizado en elementos secundarios del programa, como el fondo o elementos gráficos de menor importancia.
- **Gris claro (#E5E5E5):** El gris claro es un color suave y tranquilo que puede transmitir una sensación de calma y serenidad. Puede ser utilizado en el fondo del programa o en elementos secundarios que requieren un tono neutro y no llamar demasiado la atención.
- **Azul oscuro (#003366):** El azul oscuro es un color profundo y sereno que puede transmitir estabilidad y confianza. Es un color ideal para destacar elementos importantes del programa, como el botón de búsqueda o el campo de texto de entrada, para atraer la atención del usuario y transmitir un mensaje de confianza.
- **Azul claro (#66B2FF):** El azul claro es un color fresco y relajante que puede transmitir una sensación de calma y tranquilidad. Se puede utilizar para destacar información relevante como los resultados de búsqueda o la información de contacto, para transmitir una sensación de claridad y frescura.

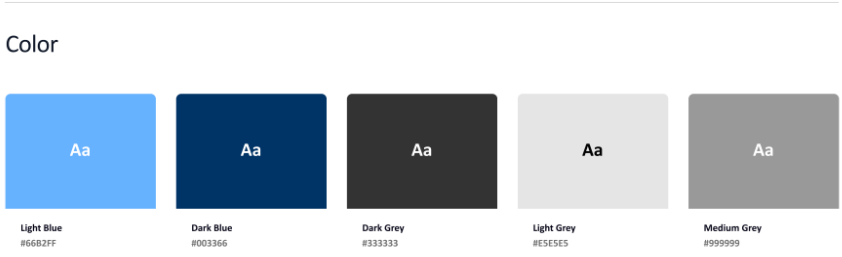

Ilustración 9 Guía de estilos para la paleta de colores

En general, la paleta de colores que se ha elegido es una combinación equilibrada que transmite seriedad, confianza y tranquilidad, lo que es ideal para un programa de búsqueda de archivos. Utilizando cada color de manera estratégica, se puede crear un

diseño atractivo y funcional que ayude al usuario a encontrar la información que necesita de manera rápida y efectiva.

6.3. Algoritmos de búsqueda

Se ha optado por implementar un algoritmo de búsqueda secuencial en lugar de los algoritmos de búsqueda secuencial indexado o de búsqueda binaria en el desarrollo del presente proyecto. Esta elección se fundamenta en la naturaleza de los sistemas de archivos desestructurados y, en particular, en su aplicación en el ámbito de las asesorías legales.

En sistemas de archivos desestructurados, la falta de una organización predefinida y la variabilidad en la disposición de los datos hacen que la búsqueda secuencial sea más eficaz. Los algoritmos de búsqueda secuencial indexado y búsqueda binaria suponen una estructura de datos más organizada, lo que no es necesariamente el caso en sistemas de archivos desestructurados. La búsqueda secuencial permite examinar cada elemento uno por uno, sin hacer suposiciones sobre la distribución de los datos, lo que se alinea mejor con la diversidad de formatos y organización que podría presentar la información en el contexto de asesorías legales.

En el ámbito de asesorías legales, la precisión y la exhaustividad en la búsqueda de información son cruciales. Los documentos legales y la información relacionada pueden estar dispersos en diferentes formatos y estructuras, lo que dificulta la implementación de algoritmos que requieren un patrón específico de organización. El algoritmo de búsqueda secuencial se adapta de manera más efectiva a esta variabilidad, ya que aborda cada elemento en su secuencia, sin asumir ninguna organización subyacente.

Además, se ha incorporado una estrategia de búsqueda en profundidad, la cual permite a los usuarios examinar minuciosamente la información relacionada con sus búsquedas. La búsqueda en profundidad implica la exploración exhaustiva de todos los archivos y carpetas, adentrándose en niveles más profundos del sistema de archivos para asegurarse de no omitir ningún documento relevante. En el contexto de asesorías legales, donde cada detalle puede tener implicaciones significativas en la toma de decisiones, la capacidad de realizar una búsqueda en profundidad se erige como un elemento esencial. Esta función garantiza que ningún archivo pertinente quede sin ser considerado,

contribuyendo así a una evaluación comprehensiva y precisa de la información en este ámbito crucial.

La elección de implementar la búsqueda en profundidad en lugar de la búsqueda en amplitud está fundamentada en la naturaleza de las asesorías legales, donde la exhaustividad y precisión en la recolección de datos son vitales. Aunque ambas estrategias ofrecen métodos efectivos para explorar un sistema de archivos, la búsqueda en profundidad se adecua más a la necesidad de no pasar por alto ningún detalle relevante en el proceso. Dado que los documentos legales pueden estar dispersos a lo largo de diferentes niveles y ubicaciones, la búsqueda en profundidad asegura una cobertura integral. Aunque en apariencia ambas estrategias pueden parecer similares, la elección de la búsqueda en profundidad refuerza la garantía de que se examina exhaustivamente cada rincón del sistema, asegurando que ningún aspecto crítico quede inadvertido.

En resumen, la elección de un algoritmo de búsqueda secuencial en lugar de opciones más estructuradas como la búsqueda secuencial indexada o búsqueda binaria se ha realizado en función de la adaptabilidad del enfoque a sistemas de archivos desestructurados, y particularmente en el contexto de asesorías legales. La implementación de una búsqueda en profundidad añade una capa adicional de eficacia y exhaustividad en la exploración de información en este entorno diverso y crítico.

6.4. Funcionalidad de expresiones regulares

Para habilitar la funcionalidad de utilizar expresiones regulares en el proyecto, se ha empleado la biblioteca "*System.Text.RegularExpressions.Regex*". Esta elección se basa en varias ventajas clave que esta biblioteca proporciona. En primer lugar, ofrece una amplia gama de opciones para manipular y buscar patrones en los datos. Su estructura optimizada asegura un rendimiento eficiente incluso en búsquedas complejas. Además, su facilidad de uso se destaca, ya que su sintaxis se asemeja a las funciones de manipulación de cadenas, lo que simplifica considerablemente su implementación. Todo esto se logra aprovechando las características del lenguaje de programación C#, lo que garantiza una integración perfecta en el proyecto y contribuye a una experiencia de desarrollo fluida.

7. Funcionalidades de la aplicación desarrollada

En la aplicación final hemos implementado las siguientes funcionalidades:

7.1. Búsqueda de expresiones literales en archivos TXT

Esta funcionalidad nos proporciona la capacidad de buscar expresiones literales en conjunto de archivos de un directorio, su utilidad radica en los usuarios que no están capacitados para usar expresiones regulares debido a su dificultad de compresión y su inexperiencia.

7.2. Búsqueda de expresiones regulares en archivos TXT

Esta funcionalidad nos proporciona la capacidad de realizar búsquedas utilizando expresiones regulares, les proporcionará a los usuarios capacitados para las mismas una gran versatilidad y eficiencia para poder realizar búsquedas e informes precisos.

7.3. Almacenamiento y carga de expresiones regulares con descripciones previas

Esta funcionalidad nos permitirá almacenar expresiones regulares recurrentes para su posterior uso y poder agilizar el tiempo de trabajo, no perder información y capacitar a los usuarios inexpertos con dichas expresiones para realizar búsquedas avanzadas y más precisas.

7.4. Almacenamientos de informes sobre las búsquedas

Esta funcionalidad nos habilitará una vía para poder almacenar los resultados de las búsquedas y poder almacenarlos para su posterior consulta y no perder la información de ese momento. Además, se ha complementado con la posibilidad de realizar un filtrado posterior que nos permitirá definir qué resultados almacenar y cuales no, permitiendo así mantener solo la información más relevante y deshacernos de aquella que no nos interese.

8. Líneas de trabajo futuras

Como líneas de trabajo futuras se podrían implementar las siguientes funcionalidades para enriquecer este proyecto y proporcionar más utilidades.

8.1. Realizar búsquedas en otros tipos de archivos como *PDFs* y Archivos *Words*

1. **Ampliar la utilidad del programa:** Mucha de la información importante se encuentra en formatos como PDF y Word. Al habilitar la búsqueda en estos tipos de archivos, nuestro programa adquerirá una versatilidad que y utilidad para los usuarios que les permitirá ampliar sus fuentes y obtener información de distintos tipos de archivos.
2. **Satisfacción del usuario:** La capacidad de buscar en diferentes tipos de archivos mejora la satisfacción del usuario al proporcionar una experiencia más completa y abarcadora. Los usuarios no tendrán que recurrir a diferentes herramientas para buscar información en distintos formatos, lo que simplificará su flujo de trabajo.
3. **Enriquecimiento del análisis:** Muchos documentos en PDF y Word contienen estructura y formato que pueden enriquecer el análisis. La posibilidad de extraer y presentar información de manera ordenada puede ayudar a los usuarios a entender mejor el contexto de sus búsquedas.
4. **Alineación con expectativas:** Los usuarios modernos esperan que las herramientas de búsqueda sean lo más completas y versátiles posible. La inclusión de la búsqueda en PDF y Word se alinea con esta expectativa y brinda una experiencia más satisfactoria.

8.2. Implementar la conectividad con una base de datos en red

Gracias a esta funcionalidad los usuarios obtendrían las siguientes ventajas:

1. **Centralización y accesibilidad:** Al almacenar los informes en una base de datos en red, los usuarios pueden acceder a los informes desde cualquier ubicación. Esto es especialmente valioso para equipos distribuidos o usuarios que trabajan en diferentes lugares físicos.

2. **Colaboración mejorada:** La base de datos en red permite a múltiples usuarios acceder y colaborar en informes compartidos. Esto promueve una colaboración más efectiva, ya que los usuarios pueden trabajar juntos en tiempo real, realizar comentarios y realizar cambios de manera conjunta.
3. **Historial de informes:** La base de datos en red puede mantener un historial de informes generados, lo que facilita el seguimiento de los cambios a lo largo del tiempo y proporciona un registro valioso de la evolución de los informes.
4. **Seguridad y respaldo:** Los sistemas de bases de datos en red suelen tener mecanismos integrados de seguridad y copias de seguridad, lo que garantiza la protección y recuperación de los informes en caso de fallos o problemas técnicos.
5. **Acceso controlado:** Puedes establecer niveles de acceso para diferentes usuarios, lo que permite controlar quién puede ver, editar y generar informes. Esto es especialmente útil para mantener la integridad de los datos y restringir el acceso no autorizado.
6. **Análisis y toma de decisiones:** La capacidad de almacenar informes en una base de datos en red permite a los usuarios realizar análisis más profundos, comparar informes a lo largo del tiempo y tomar decisiones informadas basadas en datos históricos.
7. **Rendimiento mejorado:** A medida que los informes aumentan en tamaño y cantidad, una base de datos en red puede ofrecer un rendimiento más estable y eficiente para gestionar y recuperar los datos.
8. **Escalabilidad:** La conectividad con una base de datos en red permite una escalabilidad más sencilla a medida que el programa y la cantidad de informes crecen con el tiempo.

9. Conclusiones

Este Trabajo de Fin de Grado (TFG) ha culminado en una solución que revisa las funcionalidades más necesarias de las asesorías legales, así como la realización de búsquedas con expresiones literales, búsquedas avanzadas como el uso de expresiones regulares, generación de informes y almacenamiento de expresiones para futuros eso.

El problema subyacente se origina en la complejidad de examinar y analizar datos en sistemas de archivos carentes de estructura definida, lo que es particularmente crítico en el contexto legal donde la precisión y la eficiencia son esenciales. La revisión de herramientas existentes en esta área reveló limitaciones significativas, como la falta de opciones de búsqueda avanzada y la ausencia de informes detallados.

Una contribución destacable es la función de generación de informes detallados, la cual no solo presenta una visión general de los archivos encontrados, sino que también contextualiza las coincidencias de búsqueda, proporcionando una comprensión profunda del contexto en el que se producen.

La implementación de esta solución se sustentó en tecnologías avanzadas como *Windows Presentation Foundation (WPF)* y el *framework* .NET, asegurando una interfaz de usuario intuitiva y atractiva, además de optimizar la eficiencia y el rendimiento del programa, lo que es esencial en el entorno legal.

En resumen, este TFG ha alcanzado su objetivo de desarrollar un programa de búsqueda en sistemas de archivos desestructurados que explore nuevas líneas de uso para este tipo de programas, diseñado específicamente para las asesorías legales. Con funcionalidades avanzadas de búsqueda, almacenamiento y generación de informes detallados. En consecuencia, proporciona a los profesionales legales una herramienta de fácil manejo para sus tareas diarias.

10. Referencias y bibliografía

HTML: Lenguaje de etiquetas de hipertexto | MDN. (s. f.). MDN Web Docs. Recuperado 15 de agosto de 2023, de https://developer.mozilla.org/es/docs/Web/HTML

Qué es HTML: guía sobre este lenguaje de programación básico. (s. f.). Next_U. Recuperado 15 de agosto de 2023, de https://www.nextu.com/blog/que-es-html-rc22/

XAML overview - UWP applications | Microsoft Learn. (s. f.). Microsoft. Recuperado 15 de agosto de 2023, de https://learn.microsoft.com/en-us/windows/uwp/xaml-platform/xaml-overview

Serrano, J. (s. f.). *¿Qué es WinRT? – Jorge Serrano*. Geeks. Recuperado 15 de agosto de 2023, de https://geeks.ms/jorge/2012/06/11/qu-es-metro/

C# por DesarrolloWeb.com. (s. f.). Desarrollo Web. Recuperado 15 de agosto de 2023, de https://desarrolloweb.com/

Martínez Canelo, M. (s. f.). *¿Qué es la Programación Orientada a Objetos?* Profile. Recuperado 15 de agosto de 2023, de https://profile.es/blog/que-es-la-programacion-orientada-a-objetos/

C# docs - get started, tutorials, reference. | Microsoft Learn. (s. f.). Microsoft. Recuperado 15 de agosto de 2023, de https://learn.microsoft.com/en-us/dotnet/csharp/

Getting Started - WPF .NET Framework | Microsoft Learn. (s. f.). Microsoft. Recuperado 15 de agosto de 2023, de https://learn.microsoft.com/en-us/dotnet/desktop/wpf/getting-started/?view=netframeworkdesktop-4.8

¿Por qué debes tener una guía de estilo antes de comenzar tu app .. (s. f.). 3Androides. Recuperado 16 de agosto de 2023, de https://www.3androides.com/actualidad/159-por-que-debes-tener-una-guia-de-estilo-antes-de-comenzar-tu-app-movil

¿Quién diseñó Calibri? La icónica tipografía de Microsoft. Graffica. Recuperado 16 de agosto de 2023, de https://graffica.info/quien-diseno-calibri-la-iconica-tipografia-de-microsoft/

Paredes, J. (2023, junio 1). *Figma: Qué es, para qué sirve el programa y cómo usarlo*. Escuela británica De Artes Creativas Y tecnología. Recuperado 16 de agosto de 2023, de https://ebac.mx/blog/que-es-figma

What Is Unstructured Data? Structured Data vs Unstructured | NetApp. (s. f.). NetApp. Recuperado 23 de agosto de 2023, de https://www.netapp.com/data-storage/unstructured-data/what-is-unstructured-data/

Gráficos vectoriales. (s. f.). Universidad Nacional De Quilmes. Recuperado 23 de agosto de 2023, de http://libros.uvq.edu.ar/spm/246_grficos_vectoriales.html

Ponce, J. (2021, febrero 21). *Algoritmos de búsqueda - Jahaziel Ponce*. JAHAZIEL PONCE. Recuperado 24 de agosto de 2023, de https://jahazielponce.com/algoritmos-de-busqueda/

López Mamani, M. (2020, mayo 25). *Ya utilizo Selenium, ¿por qué debería aprender Cypress?*. Encora. Recuperado 30 de agosto de 2023, de https://www.encora.com/es/blog/ya-utilizo-selenium-por-que-deberia-aprender-cypress

Regular Expression Language - Quick Reference | Microsoft Learn. (2023, marzo 10). Microsoft. Recuperado 30 de agosto de 2023, de https://learn.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference

Rehkopf, M. (s. f.). *Historias de usuario | Ejemplos y plantilla | Atlassian*. ATLASSIAN. Recuperado 4 de septiembre de 2023, de https://www.atlassian.com/es/agile/project-management/user-stories

Jimenez-Torres, V. H., Tello-Borja, W., & Rios-Patiño, J. I. (2014, diciembre 4). Redalyc.Lenguajes de Patrones de Arquitectura de Software: Una .. Universidad Tecnológica De Pereira. Recuperado 4 de septiembre de 2023, de https://www.redalyc.org/pdf/849/84933912003.pdf