

**Actividad 6.1 Análisis de herramientas usadas para la solución de las situaciones problema**

Carlos Estrada Ceballos A01638214

Natalia Velasco García A01638047

Abigail Velasco García A01638095

Eduardo Esteva Camacho A01632202

28 de mayo del 2021

Implementación de métodos computacionales

TC2037.2

La situación problema planteada en este curso fue abordado en partes, el primer paso consiste en hacer un análisis, el segundo en diseñar un plan, el tercero en ejecutarlo y por último en la etapa cuatro se valida el resultado. Para cada etapa de la solución decidimos usar ciertas herramientas de acuerdo con distintos criterios tales como accesibilidad, facilidad, eficiencia y funcionalidad.

En la etapa de análisis las herramientas que usamos tienen como fin analizar el lenguaje, puesto que la situación problema se trata del procesamiento del lenguaje, en esta categoría se encuentran los autómatas finitos deterministas, expresiones regulares y JFlap.

La siguiente etapa es la de diseño de la solución, en esta etapa es necesario documentar el plan, para esto se usaron aplicaciones para hacer diagramas, otras herramientas que se consideraron dentro de esta categoría son las que están relacionadas con paralelismo y concurrencia, pensando en mejorar la eficiencia de la solución propuesta.

Finalmente, las etapas tres y cuatro están dentro de la categoría de implementación, aquí se toman en cuenta las herramientas que nos fueron más útiles al ejecutar lo que planeamos en las etapas pasadas.

A continuación, se presentan todas las herramientas utilizadas más a detalle y las razones por las que consideramos que eran necesarias para resolver la situación problema.

**Análisis del lenguaje**

El procesamiento del lenguaje nos permite crear herramientas que facilitan muchas tareas en la vida cotidiana, por ejemplo, aplicaciones como traductores, auto correctores, buscadores, editores, compiladores, intérpretes, etc. Esto es posible gracias al estudio del lenguaje, para entenderlo es necesario hacer un análisis léxico, sintáctico y semántico. (Olmos, 2018)

Existen varios medios para procesar el lenguaje, el que se utilizó en clase fue un generador de análisis léxico, justo en eso consiste una parte de la situación problema, en hacer un programa que reconoce e identifica los tipos de tokens de un conjunto de símbolos (La entrada) y determina si pertenecen o no al lenguaje para después crear como salida un archivo que muestra cada token y su tipo.

La solución propuesta en el equipo fue determinar la categoría léxica de cada token, leer la entrada de símbolos lexema por lexema y agruparlos hasta encontrarse con un token de otra categoría que no puede unirse al grupo. Las herramientas que se usaron para resolver esa tarea son expresiones regulares, para la especificación de tokens y autómatas finitos deterministas para estructurarlos.

Las expresiones regulares son un metalenguaje, un metalenguaje sirve para describir un lenguaje, las expresiones regulares describen los elementos léxicos; orden y precedencia, repeticiones, y agrupaciones. Esto permite agrupar símbolos, concatenar cadenas y elegir alternativas, en resumen, encontrar palabras. (Cueva, 2000)

“Una máquina abstracta o autómata es un dispositivo teórico capaz de recibir y transmitir información” (Isasi, Martínez, Borrajo, 1997, p. 2). Está formado por un conjunto de estados, un estado inicial y subconjuntos de estados definidos y finales; los estados son representados por nodos; también tiene un conjunto de tokens aceptados, y una función de transición representada por arcos. (Olmos, 2018)

El nodo en el que está la cinta al leer una entrada es el estado inicial, según el símbolo de entrada ocurren las transiciones entre estados y cuando llega a un estado final se puede terminar la palabra o cambiar de estado.

Los autómatas finitos y las expresiones regulares se relacionan de forma directa, los autómatas reconocen o validan palabras, aceptan secuencias de transiciones de acuerdo con las reglas del lenguaje descritas por las expresiones regulares. (Isasi, Martínez, Borrajo, 1997).

Estas herramientas son muy importantes, más allá de ayudarnos a implementar un identificador y resaltador léxico, fueron de gran ayuda para entender cómo funcionan los lenguajes naturales y formales y cómo funcionan las herramientas de procesamiento de lenguaje que usamos todos los días.

Conocerlas es de gran utilidad en el desempeño de nuestra carrera, puesto que nos permite tener un mejor criterio para evaluar y comparar herramientas de procesamiento de lenguaje para tomar mejores decisiones y facilita adaptarse a un nuevo lenguaje de programación.

**JFlap**

JFlap es una herramienta que facilita el aprendizaje de la teoría de autómatas, esta te permite “experimentar con temas de lenguajes formales que incluyen autómatas finitos no deterministas, autómatas de pila no deterministas, máquinas de Turing de cintas múltiples, varios tipos de gramáticas, análisis sintáctico y sistemas L.” (Rodger, 2005)

Esta aplicación fue desarrollada con el objetivo de mejorar la experiencia de aprendizaje de estudiantes universitarios y de preparatoria; otorga feedback inmediato, permite a los estudiantes aplicar la teoría por lo que la complementa, al representar gráficamente los conceptos los hace más entendible, además permite realizar pruebas con diferentes inputs para conocer las limitantes de la solución y si es correcta.

Está aplicación es gratuita además su página web contiene toda la información necesaria para usar el software. Esta cuenta con un libro, módulos, ejercicios, presentaciones, documentación, videos, guías, tutoriales, artículos, etc.

**Diseño de desarrollo de software**

**Diagramas UML**

El Lenguaje unificado de modelado (UML) es indispensable para lograr un buen desarrollo de software, ya que te permite documentar el diseño y las decisiones tomadas sobre este, abordar un problema en partes, definir requerimientos, representar información para los diferentes stakeholders y aumentar la calidad del software.

Dos de las herramientas más comunes para realizar estos diagramas son Lucidchart y app.diagrams.net. Ambas satisfacen necesidades diferentes y dependiendo del proyecto una es mejor que la otra; para estudiantes app.diagrams.net sería la aplicación más conveniente por su accesibilidad, en cambio Lucidchart es una mejor opción para empresas por todas sus herramientas de colaboración y sus integraciones. Las diferencias entre ambas aplicaciones están descritas en la siguiente tabla:

|  |  |  |
| --- | --- | --- |
|  | **Lucidchart** | **app.diagrams.net** |
| Accesibilidad | - Es necesario pagar por el servicio.  - Cuenta con una versión gratuita, sin embargo, limita el total de elementos que puedes insertar en todos tus documentos.  - Puede ser utilizada en dispositivos móviles Android y iPhone. | - Es una aplicación gratuita.  - Para realizar un documento no es necesario tener una cuenta.  - Tiene una versión de escritorio disponible para Windows, macOS y Linux, esto te permite realizar diagramas sin internet. |
| Facilidad de implementación | -Cuenta con plantillas, guías y una herramienta para encontrar funciones.  - Tiene más plantillas con diseños más variados sin embargo la mayoría son para cuentas premium.  - Vincula datos de otros documentos que son actualizados automáticamente en el diagrama.  - Puede importar diagramas de otras aplicaciones. | - Todas las plantillas son gratuitas.  - Puedes agregar figuras individuales.  - La aplicación tiene un diseño intuitivo.  - Puede trabajar con diagramas realizados desde 2005 e importar diagramas de otras aplicaciones. |
| Colaboración | - Permite comentar cada elemento y editar un documento al mismo tiempo.  - Registra los cambios y los usuarios que los realizaron. | - Se colabora por medio de Google Drive.  - No registra los cambios que hace cada usuario.  - Permite comentarios. |
| Seguridad | - Cuenta con bloqueo de dominio, soporte técnico y asignación de licencias | - La empresa no puede acceder a los datos.  - Los documentos solo se guardan en plataformas de almacenamiento en la nube o en la memoria local del dispositivo |
| Integraciones | G Suite, Microsoft Office, Slack, Jira, Confluence, Salesforce, Slack, AWS, Azure, GCP, Zapier, GitHub, BambooHR, Quip, LeanIX, Asana, AuditBoard, Nextpond, UVexplorer, Zoho Connect, Nuclino y Dropbox Paper. | Atlassian Confluence, Jira, Google Drive, OneDrive, GitHub, Aha!, Atom, BookStack, Docstell, FOSWiki, Grafana, Growi, JupyterLab, LumApps, Nextcloud, Nuclino, ownCloud, Redmine, Samepage, Sandstorm, Tiki Wiki CMS Groupware, VS Code y XWiki. |

**Paralelismo y concurrencia**

Las últimas herramientas que utilizamos en el transcurso del semestre se basaron en el paralelismo, o programación concurrente. Sin embargo, a pesar de lo que indica el nombre, esto no es reflejo de la manera en la que se programa, sino de la manera en la que se computa. Por lo general, en el transcurso de la carrera hemos programado código que utiliza procesamiento serial. Es decir, el código se ejecuta una instrucción a la vez de manera secuencial. Un solo procesador de la computadora carga una por una cada instrucción. Sin embargo, al manejar cantidades masivas de datos, puede ser más eficiente utilizar la programación concurrente. Utilizando esta herramienta, se utilizan varios procesadores al mismo tiempo, reduciendo la carga de trabajo por cada procesador que se utiliza y, por lo general, reduciendo el tiempo requerido para correr el programa.

Si comparamos el uso de ambas herramientas, el procesamiento serial es más fácil de programar. Es la manera en la que uno se acostumbra a programar cuando tiene poca experiencia y puede ser generado de manera más rápida el código. En cambio, a pesar de ser una herramienta más compleja en su uso, puede acelerar de manera drástica la velocidad y eficiencia del programa. En efecto, si se utiliza la notación “Big O,” se puede observar un ejemplo muy simple, propuesto por S. Rastogi and H. Zaheer. (2016). En este ejemplo, un programa con complejidad O(n), donde ‘n’ es igual a la cantidad de procesos independientes en el programa, se puede reducir a O(1) si se distribuyen los procesos en ‘n’ procesadores. Si visualizamos la reducción de complejidad O(n) a O(1) es posible apreciar cuánto puede reducir la carga de trabajo esta herramienta.

Adicionalmente, Barney, B (s.f.) propone que el procesamiento concurrente está mejor adaptado para procesar cantidades enormes de datos acerca de cómo opera el mundo. Él propone que la computación paralela es una herramienta útil para el modelado y la predicción de movimientos planetarios, fenómenos meteorológicos, algo tan simples como el tráfico en una ciudad y mucho más. Esto se debe a que el análisis de estos fenómenos es en realidad el análisis de varios eventos simultáneos sin conexión, algo que va muy en línea con la programación paralela.

Barney, B (s.f.) también propone que el procesamiento paralelo es el futuro de la informática. Actualmente se considera estándar vender para uso personal computadoras con múltiples procesadores, y a través de esta herramienta nos estamos acercando a nuevas alturas en el procesamiento de datos. También D’Agostino, D, Kotenko, I., Liò, P., &. Merelli, I. (2020) observan como el paralelismo empezó como una manera compleja de programar (que tenía mucha promesa), actualmente se considera ideal que cualquier programador se familiarize con la arquitectura detrás del procesamiento concurrente. Es por esto, que a pesar de la simplicidad del cómputo serial, se usará procesamiento paralelo para obtener un mejor resultado en la entrega final del proyecto.

Una manera de poner en acción en el paralelismo el uso de hilos. Un hilo es una parte pequeña del proceso que ejecuta un código. Como observa Wilson, C (2019) en una computadora con un solo procesador, se tendría que elegir qué hilo correr en cada dado momento. Cambiar de actividad requeriría pausar un hilo y empezar otro, lo cual es útil, pero no es verdadero multitasking. En cambio, en computadoras modernas con múltiples procesadores, cada procesador puede ocuparse de un hilo, resultando en mayor eficiencia. Estos procesos los exploramos en el lenguaje de programación Java, ya que la creación de hilos es parte de la biblioteca estándar de Java y ofrecía una manera conveniente de explorar el tema.

Además, Java ofrece a través de las herramientas en su kit de desarrollo la oportunidad de desarrollar GUI (graphic user interface) para que un programa se ejecute de manera visual sin depender de la consola, algo que deja más claro los resultados del programa que se desarrolla. De esta manera se puede controlar un poco mejor el input de los usuarios para evitar errores y asegurar que haya claridad en la consecuencia de cada acción de los usuarios. Para el aprendizaje del uso de GUIs y la creación de hilos, Java es una herramienta ideal, ya que ofrece herramientas para llevar esto a cabo de manera simple sin configuración adicional. En cambio, en un lenguaje de programación como C++, en el que tenemos más experiencia, no es parte de la librería estándar el acceso a métodos para manejar hilos o desarrollar GUIs. Por lo tanto optamos por utilizar Java para explorar las posibilidades de la creación de hilos y GUIs, para poder prepararnos para aplicar lo aprendido en otros lenguajes de programación.

Sin embargo, cabe notar que debido a la naturaleza del programa final que se desarrollará para esta materia, no hará falta generar GUI. Por lo tanto, simplemente aprender a crear hilos en C++ será lo único requerido para generar un producto final exitoso. Sin embargo, si fueran necesario los GUIs para este proyecto, tal vez sería de más conveniente y eficiente simplemente adaptar el código que tenemos a Java que buscar librerías externas que se adapten a nuestras en C++.

**Trabajo colaborativo**

**GitHub**

Github como definición es un sistema de control de versiones de software, es uno de los sistemas más utilizados en todo el mundo así como también el más popular al momento de querer manejar las distintas versiones del código fuente desarrollado por un equipo de trabajo, sin embargo, GitHub no es la única opción al momento de realizar estas tareas, dentro de este campo también existe GitLab, la cual es una gran herramienta como opción a GitHub, como su nombre lo indica en ambas opciones, están basadas en el software de versiones de Git.

Aunque son herramientas muy parecidas que parecen cumplir los mismos objetivos tienen ciertas diferencias las cuales hacen que los usuarios se inclinen más hacia una herramienta que la otra, por ejemplo, en el caso de GitLab durante mucho tiempo tuvo la gran ventaja de ofrecer infinitos repositorios gratuitos a sus usuarios, GitHub observó que esto parecía ser una desventaja importante y decidieron de igual manera ofrecer esta característica a sus usuarios.

Aun con esto, se puede observar que GitHub sigue contando con más restricciones en su versión gratuita en comparación con GitLab, entre otras ventajas de GitLab también se encuentra que en general, GitHub suele ofrecer menos derechos de usuario, mientras que GitLab ofrece una gestión de derechos con varios roles por defecto, GitHub por su parte solicita una suscripción de pago para poder tener acceso a una función parecida.

En nuestro caso decidimos utilizar GitHub por ciertas razones y ventajas con las que cuenta, gracias a la enorme comunidad con la que cuenta GitHub aumenta las posibilidades de encontrar colaboradores para un proyecto propio, sobre todo, en el ámbito del código abierto. Además de lo anterior, la integración de repositorios de otros usuarios dentro de GitHub es más sencilla. Es la plataforma que tiene más desarrolladores trabajando en ella por lo que constantemente se está actualizando. Es por esto que GitHub es considerada como la plataforma más estable y potente.

Además de todas las ventajas mencionadas anteriormente todos los integrantes del equipo contábamos con experiencia trabajando dentro GitHub, lo cual no habíamos hecho con GitLab, esto también fue un factor importante al momento de decidirnos por usar esta herramienta.

**Implementación del resaltador de sintaxis**

**C++**

Dentro del desarrollo de software existen muchas opciones hacia donde puedes orientarte, una de las principales cosas sobre las que tenemos que decidir es el lenguaje de programación que usaremos para cada caso o problemática a abordar, hoy en día los lenguajes de programación más populares para un propósito general son C, C++, Java o Python, nos gustaría hacer la comparativa entre C++ el cual es lenguaje que elegimos para la actividad y Java, el cual es un lenguaje muy usado desde hace ya mucho tiempo y que al igual a C++ es bastante estable.

Como se mencionó anteriormente Java es un lenguaje usado desde hace mucho tiempo, se hizo muy popular y muy usado por grandes compañías alrededor del mundo debido a las grandes ventajas que este ofrece, una de estas ventajas es que corre dentro de su máquina virtual lo cual hace que tenga una gran adaptabilidad y universalidad dentro de cualquier sistema operativo, de ahí su lema “write once, run everywhere”, sin embargo, esto muchas veces juega en su contra ya que pierde eficiencia, también en tiempos anteriores este lenguaje solía ser muy criticado por su velocidad, sin embargo en versiones más recientes han logrado disminuir considerablemente esta problemática al punto de llegar a ser uno de los lenguajes más rápidos, y por último una de las características más importantes es que te permite acceder a la programación paralela de una manera muy sencilla, esto con la creación de hilos mediante la herencia de la clase Thread.

Por otro lado tenemos a C++, que de igual manera que con Java es un lenguaje muy antiguo, este lenguaje es una extensión de C con la intención de crear un lenguaje con propósitos más generales y no tan específicos, su quizás mejor punto a favor son sus casi inigualables niveles de desempeño. Además de esto C++ es ligero, cuenta con gestión manual de memoria y de igual manera se puede usar en cualquier plataforma existente, quizás una de sus principales desventajas sea que debido a las libertades de gestión de memoria que otorga es que muchas veces suele tener ahí sus puntos en contra, muchas de estas funciones deben de ser creadas por los mismos desarrolladores para cuidar esta parte  lo cual no siempre es lo mejor para el usuario.

Tomando en cuenta puntos como los anteriormente mencionados tomamos la decisión de usar C++ basándonos en 3 cosas principalmente, la primera es que cuenta con un desempeño casi inigualable, esto hace que sea muy rápido el momento de ejecución y de realización de tareas especificadas previamente, otro punto importante es que no es tan estricto como Java, cuenta con una sintaxis mucho más simple que ayuda a la fluidez al momento de la codificación, esto se ve muchas veces en que en Java se producen errores muy rígidos los cuales no existen en C++, y por último tenemos que todos nosotros hemos trabajado con este lenguaje, estamos muy familiarizados con él y nos es más fácil y ágil el desarrollo de software en equipo, mientras que en Java apenas estamos dando nuestros primeros pasos y no contamos con el conocimiento de la sintaxis ni experiencia de desarrollo necesaria dentro de este lenguaje.

**HTML y CSS**

Para poder presentar los resultados de nuestro análisis dentro de nuestro archivo necesitamos generar una salida que el usuario pueda visualizar, para esto usamos HTML, para poder presentar la información como tal, así como CSS para poder darle un estilo al texto o información que se genere. Aunque HTML y CSS son la manera más conocida para presentar información de manera estilizada, también existen alternativas a esto como lo puede ser XML, la sintaxis de XML es muy parecida a la de HTML aunque cada una con sus respectivas características.

Para poder realizar una buena comparativa entre ambas, debemos de comenzar a entender qué es y qué implica cada una de ellas, XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

Este lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas, lo cual es precisamente lo que generamos dentro de nuestro análisis. XML nos proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado, en pocas palabras, nos puede ayudar a presentar los resultados de nuestro análisis lexicográfico de una manera personalizada.

Por otro lado tenemos al ya conocido HTML que de la mano de CSS nos logran dar resultados increíbles de acuerdo a cada necesidad que estemos buscando, HTML o Hypertext Markup Language es el lenguaje de marcado de documentos por defecto para construir páginas web, los comandos de formato utilizados en los contenidos para web se refieren a la estructura del mismo y al diseño que se mostrará en el navegador.

En conjunto tenemos a CSS el cual nos permite darle estilos a la información contenida dentro de nuestro documento HTML, gracias a la diversidad de elementos y atributos que tiene CSS nos permite crear prácticamente cualquier diseño que tengamos en mente.

A pesar de los beneficios que nos ofrece XML como lo puede ser el énfasis en la presentación de datos, nosotros optamos por usar HTML de la mano de CSS, una de las razones es por la facilidad de poder observar los resultados, gracias a que se utiliza en el navegador, podemos ver los resultados estéticos simplemente abriendo el archivo con cualquier navegador que tengamos dentro de nuestra computadora, los resultados pueden ser observados desde prácticamente cualquier cualquier dispositivo.

Otra de las razones es que HTML es mucho más flexible con su sintaxis, en el caso de tener algún error en la creación del documento, el propio navegador intenta corregir estos errores de la manera que el considere correcto, por ejemplo, en XML si no se cierra alguna de las tag previamente abierta te va a marcar un error, sin embargo en HTML, es posible que el navegador se encargue de corregirlo y que se siga observando de una manera correcta.

Por último el punto de la experiencia, todos estamos familiarizados con estos lenguajes, al dominar ambas cosas tanto HTML como CSS nos permite realizar la actividad de una manera más fluida y que nos sea muy fácil asignar las características de estilos al resultado generado.

**IDE**

    Los IDEs utilizados en el desarrollo del resaltador de sintaxis fueron Visual Studio Code y NetBeans, el primero para la implementación del resaltador y el segundo para manejar hilos.

    Visual Studio Code es un software gratuito y ligero que soporta múltiples lenguajes gracias a sus extensiones, tiene una extensa documentación y actualizaciones mensuales. Esta herramienta fue elegida por estas razones, también por la familiaridad que los integrantes del equipo tenemos con el IDE, además cuenta con extensiones que facilitan el trabajo colaborativo cómo Live Share, esta herramienta te permite desarrollar software en tiempo real con las personas con las que compartas el proyecto.

NetBeans es un IDE de Oracle, lo consideramos como la mejor opción porque es gratuito, multiplataforma, fácil de usar, soporta Java y por su interfaz gráfica.

**Tecnologías Emergentes**

**Análisis del lenguaje**

|  |  |
| --- | --- |
| **App** | **Año** |
| MTSolution | 2003 |
| Grafos | 2003 |
| AtoCC | 2004 |
| GraphTing | 2001 |
| JFlap | 2018 |

Estas herramientas modelan relaciones matemáticas, tienen propósitos educativos y la mayoría se enfoca en teoría de lenguaje y autómatas. Fue complicado encontrar apps de teorías de autómatas porque muchas de ellas están descontinuadas, tampoco encontré ningún proyecto en desarrollo de este tipo, en conclusión no se desarrollan estas herramientas con mucha frecuencia.

**Diseño de desarrollo de software**

En esta área la frecuencia de aparición de nuevas tecnologías no es tan alta, ya que todas estas resuelven la misma necesidad; facilitan la creación de diagramas, el manejo de proyectos y permite trabajo Table

Description automatically generatedcolaborativo. Todos estos diagramas tienen estándares que todas estas aplicaciones tienen que seguir, sin embargo se actualizan constantemente y obtienen nuevas integraciones.

Alrededor de cada dos años se crean nuevas tecnología en esta área por los diferentes sistemas operativos que soportan ciertas de las aplicaciones, porque las necesidades de una empresa y de un individuo son diferentes, por los diferentes dispositivos en las que se puede usar la aplicación, diversas curvas de aprendizaje y el mercado al que se dirigen (como expertos o principiantes).

**Paralelismo y concurrencia**

Las nuevas tecnologías aparecen con frecuencia aproximada de cada par años, como lo dicta la ley de Moore “La cantidad de transistores que puede poner en un chip de computadora se duplica (aproximadamente) cada dos años.” (Walker, s.f., diapositiva 9). Esto es posible gracias al rápido desarrollo de hardware, sin embargo tiene limitaciones; el calor, la disipación de poder son las causantes de que no valga la pena hacer aumentar la velocidad de los procesadores. El paralelismo y la concurrencia son una solución para este problema, ya que es posible agregar muchos CPU a chips y así una computadora puede realizar sus tareas y ejecutar procesos con mayor velocidad..

**Implementación del resaltador de sintaxis**

    La frecuencia con la que aparecen nuevas tecnologías en esta área es muy alta por el constante cambio de hardware, por todas las necesidades nuevas que surgen a lo largo del tiempo y porque siempre se puede facilitar y mejorar el desarrollo de software.

    Surgen nuevos lenguajes de programación con mucha frecuencia por el progreso de la tecnología. Al principio las computadoras funcionaban con inputs manuales, después existieron los lenguajes de alto y bajo nivel, surgieron los compiladores, diferentes paradigmas, internet, las bases de datos, interfaces gráficas de usuario (GUI) y aplicaciones web estáticas y dinámicas. (Lestal, 2020)

Los IDE aumentaron la productividad de los programadores, ya que antes de su existencia el código se desarrollaba en un editor de texto, después se compilaba y los errores no estaban vinculados al código el programador tenía que apuntarlos y corregir su código. Los compiladores permitieron revisar el código en tiempo real y vincular los errores con links, implementaron el desarrollo visual, objetos prefabricados que puedes arrastrar y soltar, interfaces, computación en la nube, implementación, monitoreo y colaboración. ( Mendix, 2013)

 Cada lenguaje y cada IDE permite a los desarrolladores desarrollar software que funcione, de calidad, de manera eficiente dependiendo de sus necesidades.

**Conclusión**

Considerando toda esta tecnología emergente en diversos campos que contribuyeron al desarrollo del proyecto, es importante observar el impacto del pensamiento formal en el desarrollo de dichas contribuciones. El proceso del pensamiento formal es uno basado en análisis, abstracción y resolución de problemas. Este tipo de pensamiento nos ofrece la capacidad de desarrollar teorías e hipótesis, así como los medios para comprobar la validez de cualquier propuesta a la que lleguemos debido al pensamiento abstracto, a través del análisis. Es decir, es la base del método científico.

En el ambiente actual de la informática, la cantidad de información que se tiene que procesar gracias al Internet ha contribuido a una búsqueda de procesamiento de información más rápido y eficiente para cantidades masivas de datos. Con desarrollo constante y rápido en hardware y un aumento acelerado en los datos que se deben procesar para ciertas industrias, resultan necesarias las innovaciones en el campo del software. Dado que el software no puede avanzar al mismo ritmo que estos otros factores, resulta necesario llevar a cabo soluciones que involucran usos diferentes herramientas existentes. Por ejemplo, ejecutar un programa de manera iterativa en vez de recursiva no es un avance o un avance tecnológico, simplemente es un uso diferente de herramientas preexistentes.

De esta manera, a través de análisis se pueden identificar prácticas de programación que no son eficientes, y abstraer nuevas maneras de abordar el mismo problema. De esta manera, se podría decir que el pensamiento formal fomenta la optimización de herramientas en el campo de software, lo cual beneficia a todos trabajando en el campo, e inspira a otros a buscar soluciones alternativas usando el mismo proceso, generando así un ciclo virtuoso.

**Liga de repositorio**

[**Actividad\_6.1**](https://github.com/CarlosChivas/ChivasLacticas/blob/master/Actividad%206.1.docx)

**Referencias**

Barney, B. (s.f.). Introduction to Parallel Computing Tutorial. *Lawrence Livemore National Laboratory*. <https://hpc.llnl.gov/training/tutorials/introduction-parallel-computing-tutorial>

Bustos, A. J. (2018 de Septiembre de 16). OpenWebinars. Por qué aprender C++.Recuperado el 28 de Mayo de 2021, de  <https://openwebinars.net/blog/por-que-aprender-c/>

D’Agostino, D, Kotenko, I., Liò, P., &. Merelli, I. (2020). Latest advances in parallel, distributed,

and network‐based processing. *Concurrency and Computation: Practice and Experience*. Recuperado de <https://doi.org/10.1002/cpe.5683>

diagrams.net. (10 de marzo de 2019). *Your diagram data is secure and private*. diagrams.net <https://www.diagrams.net/blog/data-protection>

diagrams.net. (27 de enero de 2020). *How to use diagrams in Google Docs*. diagrams.net. <https://www.diagrams.net/#pills-three-code-features>

 diagrams.net. (s.f.). *diagrams.net Integrations*. diagrams.net. <https://www.diagrams.net/integrations>

‌Ionos. (6 de Octubre de 2020). Ionos. GitLab vs. GitHub: comparativa de ambos sistemas de control de versiones. .Recuperado el 27 de Mayo de 2021,  de <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/gitlab-vs-github/>

Lestal, J. (21 de octubre del 2020). History of programming languages. DevSkiller. https://devskiller.com/history-of-programming-languages/

Lucidchart. (s.f) *Integraciones con Lucidchart.* Lucidchart. <https://www.lucidchart.com/pages/es/integraciones>

Mendix. (4 de febrero de 2013). *The History of Visual Development Environments: Imagine There’s no IDEs. It’s Difficult if You Try.* mendix. <http://avellano.fis.usal.es/~lalonso/procesadores/10_conceptos_basicos_uniovi.pdf>

Microsoft 365 Team. (24 de septiembre de 2019). *La guía sencilla para la diagramación de UML y el modelado de la base de datos*. Microsoft. <https://www.microsoft.com/es-mx/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling>

M, L. (5 de Enero de 2021). BitDegree. Java vs C++. Recuperado el 28 de Mayo de 2021,

de <https://es.bitdegree.org/tutoriales/c-plus-plus-tutorial/>

Olmos, L.(2018).Autómatas finitos, análisis morfológico y etiquetado sintáctico <https://www.researchgate.net/profile/Lara-Olmos-Camarena/publication/330346504_Automatas_finitos_analisis_morfologico_y_etiquetado_sintactico/links/5dfbc64092851c83648af0f9/Automatas-finitos-analisis-morfologico-y-etiquetado-sintactico.pdf>

Oracle. (s.f.). *La forma más inteligente y rápida de programar.* Oracle. <https://www.oracle.com/mx/tools/technologies/netbeans-ide.html>

Rastogi, S. & Zaheer, H (2016). Significance of parallel computation over serial computation.

*IEEE Xplore*. Recuperado de <https://ieeexplore.ieee.org/document/7755106>

Rodger, S. (1 de julio de 2005). *JFLAP Version 7.1 RELEASED July 27, 2018.* JFLAP. <http://www.jflap.org/>

Rodger, S. (2007). *JFLAP - Software for Experimenting with Formal Languages*. Jflap. <http://www.jflap.org/needsfinal07.pdf>

Souza, I. d. (12 de Julio de 2019). RockContent. Recuperado el 28 de Mayo de 2021,

de [https://rockcontent.com/es/blog/](https://rockcontent.com/es/blog/que-es-xml/#:~:text=Detr%C3%A1s%20del%20dise%C3%B1o%20y%20el,para%20la%20codificaci%C3%B3n%20de%20documentos.)

Visual Studio Code. (s.f.). *Getting Started*. Visual Studio Code. <https://code.visualstudio.com/docs>

Walker, D. (s.f.). *Parallelism and Concurrency* [Diapositiva de PowerPoint]. <https://www.cs.princeton.edu/~dpw/courses/cos326-12/lec/15-parallel-intro.pdf>

Wilson, C. (2019). Multithreading and Concurrency Fundamentals. *Educative.io*. Recuperado de

<https://www.educative.io/blog/multithreading-and-concurrency-fundamentals>

Isai, V.,P., Martínez, F,P., Bojarro, M.,D. (1997). Lenguajes, gramáticas y autómatas: un enfoque práctico . Madrid, España: Addison-Wesley Iberoamericana. Recuperado de <https://books.google.com.mx/books?hl=es&lr=&id=sfzrzxMVVMUC&oi=fnd&pg=PA2&ots=W5zVYmyep3&sig=J-6YJ0iUqlZoue9tGwQ_A0952_w&redir_esc=y#v=onepage&q&f=false>