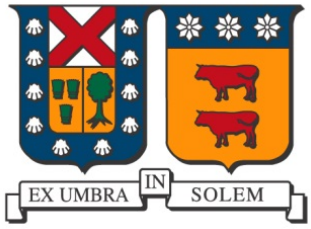
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“GENERACIÓN AUTOMÁTICA DE ARCHIVOS EN LATEX A TRAVÉS DE UNA PLATAFORMA QUE RECIBA INFORMES ESCRITOS EN FORMATO XML”

ERIC ALEJANDRO DURANTE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Xavier Bonnaire
Profesor Correferente: Nicolás Rojas

Junio - 2024

DEDICATORIA

Esto está dedicado para cada una de las personas que me acompañaron durante este largo proceso, que me aportaron dandome buenos momentos y me acompañaron en los peores pudiendo ayudarme con todo lo que tenian a su alcance, de verdad muchas gracias.

AGRADECIMIENTOS

Me gustaría agradecer a cada una de las personas que estuvieron conmigo en este largo proceso, que sin su ayuda, todo hubiera sido mucho más difícil.

A Bruno por ser un buen amigo y compañero con el que muchas veces estudiamos juntos para poder superar los desafíos que nos entregaba la carrera.

A Avril por ser una buena y apañadora novia, que me ayudó bastante a estudiar para el examen de inglés que requería al finalizar la carrera.

A mi Tío Abuelo Carlos, por siempre darme buenos consejos y apañarme en los momentos en los que más imposible parecía superar los problemas.

A mis familiares que estuvieron conmigo dándome ánimos para poder seguir, abuelos, tíos, hermana y madre, a pesar de todas las problemas que tuve.

A mi mejor amigo Manuel y su madre que sin duda me ayudaron y acogieron bastante en un momento en el que necesitaba demasiada ayuda.

A mi mejor amigo Daniel que varias veces, por muy desanimado que estaba, siempre me daba ánimos y trataba de que pudiera seguir a pesar de todas las dificultades.

A mis compañeros de carrera con los que pasé buenos momentos estudiando, realizando trabajos en grupo y recreaciones.

A Franco y Noah, que gracias a ustedes por sus consejos y por darme una manito para ganar dinero pude mantenerme en mi proceso de estudio.

Sin nada más que decir, quiero agradecerles a cada uno de ustedes por estar en los momentos más difíciles.

RESUMEN

Resumen— Muchos software de ciberseguridad generan informes de análisis forenses que resultan como formato de salida un archivo autocontenido en XML. Este formato de archivo es adecuado, dada su variedad de tags o etiquetas que tiene, para identificar, generar y almacenar resultados, además de la comunicación entre distintas plataformas, pero lamentablemente esto es nulamente práctico, debido a que es muy difícil de leer como para obtener un informe impreso.

Todo esto en consecuencia se hace necesario una plataforma web para poder traducir todo el contenido de este a archivo a código LaTeX, ya que permite una redactar fácilmente una estructuración a través de distintas secciones facilitando incluso su edición y orden, para que en cualquier caso pueda ser editable y que posteriormente podrá ser convertido en formato PDF, para que se puedan generar informes impresos y su lectura pueda ser leída al ojo humano.

Palabras Clave— XML, laTeX, traducción, tags, análisis forense

ABSTRACT

***Abstract*—** Many cybersecurity software generate forensic analysis reports that output a self-contained XML file format. This file format is suitable, given its variety of tags or labels, for identifying, generating, and storing results, as well as facilitating communication between different platforms. However, unfortunately, this is impractical, as it is very difficult to read for obtaining a printed report.

All of this consequently necessitates a web platform to translate all the contents of this file into LaTeX code, as it allows for easy structuring through different sections, facilitating even its editing and ordering, so that in any case, it can be editable and subsequently converted into PDF format, enabling the generation of printed reports and readability for human eyes.

***Keywords*—** XML, LaTeX, translation, tags, forensic analysis

GLOSARIO

Aquí se deben colocar las siglas mencionadas en el trabajo y su explicación, por orden alfabético. Por ejemplo:

DI: Departamento de Informática.

UTFSM: Universidad Técnica Federico Santa María.

PDI: Policía De Investigaciones.

XML: eXtensible Markup Lenguaje.

HTML: Hipertext Markup Lenguaje.

MTV: Model Template View.

CSS: Cascading Style Sheets.

MB: MegaBytes.

KB: KiloBytes.

ISO: International Organization for Standardization.

USB: Universal Serial Bus.

PDF: Portable.

GB: Gigabyte.

AMD: Advanced Micro Devices.

HDD: Hard Disk Drive.

CRUD: Create, Read, Update y Delete.

ÍNDICE DE CONTENIDOS

RESUMEN iv

ABSTRACT iv

GLOSARIO v

ÍNDICE DE FIGURAS viii

ÍNDICE DE TABLAS ix

INTRODUCCIÓN 1

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA 2
1.1 Contexto . . . 2
1.2 Organización . . . 3
1.3 Situación Actual . . . 4
1.4 Árbol del problema . . . 7
1.5 Objetivos . . . 8
1.5.1 Objetivo General . . . 8
1.5.2 Objetivos Específicos . . . 8
1.6 Alcance . . . 9

CAPÍTULO 2: MARCO CONCEPTUAL 10
2.1 Fundamentos Teóricos . . . 10
2.1.1 Ánalisis forense . . . 10
2.1.2 XML . . . 11
2.1.3 PDF . . . 14

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN 17
3.1 Metodología de trabajo . . . 17
3.2 Antecedentes . . . 18
3.3 Modelo de la solución . . . 19
3.4 Tecnología usada . . . 21
3.5 Frameworks . . . 23
3.5.1 NodeJS . . . 24
3.5.2 Spring Boot . . . 28
3.5.3 Gin . . . 32
3.5.4 Django . . . 36
3.5.5 Ruby On rails . . . 40
3.6 Elección del framework . . . 43
3.7 Historial de funcionamiento . . . 44
3.7.1 ¿Por qué almacenar historial de uso? . . . 44
3.7.2 Almacenamiento de la información . . . 45

3.7.3 Bases de datos 46
3.8 Trabajos en segundo plano para el procesamiento de archivos. 49
3.9 Explicación del algoritmo 50
3.9.1 Visión general 50
3.9.2 Algoritmo previo a traducción 51
3.9.3 Algoritmo de traducción 53
3.10 Tiempo de procesamiento 58

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN **61**
4.1 Introducción 61
4.2 Pruebas unitarias 62
4.2.1 Clase Translation 62
4.2.2 Clase Translator 65
4.2.3 Resultados de las pruebas unitarias. 65
4.3 Pruebas manuales 66
4.3.1 Prueba con múltiples archivos XML 66
4.3.2 Prueba con múltiples archivos XML y un archivo ZIP 72

CAPÍTULO 5: CONCLUSIONES **76**
5.1 Conclusión general 76
5.2 Cumplimiento de los objetivos 77
5.3 Trabajo a futuro 78
5.4 Palabras finales del autor 78

REFERENCIAS BIBLIOGRÁFICAS **80**

ÍNDICE DE FIGURAS

1 Gráfico de Grooming en 2019 . . . 3
2 Ejemplo de archivo XML (izquierda) y su traducción a PDF (derecha). . . . 6
3 Ejemplo de archivo TEX transformado (izquierda) y conversión a PDF (derecha). 7
4 Árbol del problema. . . . 8
5 Ejemplo de archivo HTML. . . . 13
6 Ejemplo de archivo XML. . . . 13
7 Ejemplo de un buen archivo PDF. . . . 15
8 Ejemplo de un buen archivo PDF. . . . 16
9 Modelo de la solución . . . 19
10 Página principal de la plataforma . . . 21
11 Ejemplo desarrollo en Wix . . . 22
12 Ejemplo desarrollo en softr . . . 22
13 Popularidad de NodeJS . . . 25
14 Programando en NodeJS . . . 28
15 Ejemplo de suma en código Java . . . 29
16 Syntax Java vs Python . . . 31
17 Ejemplo de una consulta GET en Spring Boot . . . 32
18 Contruyendo un endpoint que devuelve un mensaje en Gin . . . 33
19 Realizando una suma en golang . . . 34
20 Ejemplo codigo en Django . . . 37
21 Ejemplo de tipificado estático . . . 38
22 Ejemplo de tipificado dinámico . . . 38
23 Ejemplo codigo en Rails . . . 41
24 Sumando 2 numeros con clases en Ruby . . . 42
25 Bases de datos mas populares en el mundo . . . 47
26 Decompress ZIP File, Translation Class . . . 51
27 Translate Files, Translation Class . . . 52
28 Remove Files Translation Class . . . 52
29 Which template to use?, Translator Class . . . 53
30 Process XML Files, Translator Class . . . 54
31 Read TXT Format, Translator Class . . . 54
32 Translation XML To Latex, Translator Class . . . 55
33 Cleaning XML, Translator Class . . . 56
34 Create Tex File, Translator Class . . . 57
35 Create PDF and ZIP, Translator Class . . . 58
36 Gráfica de ejecución . . . 60
37 Guardar en BD sin txt . . . 62
38 Guardar en BD sin XML . . . 63
39 Guardar en BD con txt, XML y Zip . . . 63
40 Encolar un Job . . . 64
41 Encolar un Job . . . 64
42 Encolar un Job . . . 65

43 Resultados Tests 66
44 Archivo de traducción para XML 67
45 Archivo 1 XML 68
46 Archivo 2 XML 69
47 Página principal de Lifyx 69
48 Muestra de estado de conversión 70
49 Conversión completada con éxito 70
50 PDF Compilado 71
51 Archivos ZIP descargados 71
52 Plantilla ZIP del usuario 72
53 Interfaz de Lifyx con los 3 archivos 73
54 Muestra de estado de conversión utilizando archivo ZIP 73
55 Conversión completada con éxito utilizando archivo ZIP 74
56 PDF con ZIP del usuario compilado 74
57 Archivo ZIP del usuario descargado 75

ÍNDICE DE TABLAS

1 Ventajas y desventajas de Node.js 27
2 Ventajas y desventajas de Spring Boot 30
3 Ventajas y desventajas de Gin 35
4 Ventajas y desventajas de Django 39
5 Ventajas y desventajas de Rails 43
6 Tiempo de ejecución de la plataforma 59

INTRODUCCIÓN

En la era digital actual, existe mucha información que puede ser representada en diferentes y variados tipos de archivos con el objetivo de transmitir información de un sitio a otro. La difusión del contenido con fines informativos ha experimentado una expansión significativa. En este contexto, los archivos XML (Extensible Markup Language) han emergido como una herramienta fundamental para la representación y estructuración de datos complejos en una amplia gama de disciplinas. Sin embargo, el problema que tienen este tipo de archivos es su legibilidad, pudiendo dificultar la representación de información hacia otras personas, más aún cuando es usada por instituciones de carácter importante como lo son las instituciones policiales, donde algunas unidades están destinadas a la búsqueda de información que es tratada como evidencia. Por otro lado, LaTeX se ha consolidado como el estándar de facto para la composición de documentos científicos y técnicos debido a su capacidad para generar resultados de alta calidad tipográfica.

Sin embargo, a pesar de las muchas ventajas que ofrecen los formatos tanto XML como LaTeX, la transición de contenido desde el primero al segundo puede ser un proceso tedioso y propenso a errores, especialmente cuando se trata de documentos extensos o con estructuras complejas que no son legibles al ojo humano. En este sentido, la automatización de este proceso mediante una plataforma web adquiere una relevancia significativa, al simplificar y agilizar la conversión de archivos XML a LaTeX, permitiendo a las instituciones gubernamentales, como lo es la policía de investigaciones, ahorrar tiempo y enfocarse en el contenido en lugar de las tareas técnicas asociadas con la maquetación y el formateo, que muchas veces pueden ocasionar una pérdida de trabajo y tiempo.

En este trabajo, se propone el desarrollo de una plataforma web dedicada a la traducción de archivos XML a LaTeX, abarcando un área de trabajo situada en el campo de la informática, ingeniería de software y el procesamiento del lenguaje natural, con el objetivo de proporcionar una herramienta intuitiva y eficiente para la creación de documentos legibles que puedan ser presentados como evidencia oficial hacia otras personas. A través de una metodología basada en el refinamiento iterativo y la detección de casos límite, primero se parte por una propuesta base que será usada para ser probada iterativamente utilizando diferentes herramientas disponibles, lo que llevará a la plataforma web a su perfeccionamiento.

Se realizará un trabajo organizado en diferentes etapas para el desarrollo de la plataforma. En la primera etapa se llevará a cabo un análisis exhaustivo de las características y contexto del problema. Para la siguiente etapa se hará un análisis exhaustivo de las tecnologías y herramientas más adecuadas para la implementación de la plataforma. Posteriormente, se explorarán diferentes enfoques algorítmicos y técnicas de procesamiento de datos con el fin de optimizar la precisión y la eficiencia del proceso de traducción. Finalmente, se evaluará la plataforma desarrollada mediante estudios de usabilidad y pruebas de rendimiento, con el objetivo de validar su utilidad y viabilidad.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. Contexto

Las instituciones gubernamentales de carácter policial en el día de hoy realizan diversas investigaciones para los distintos delitos que ocurren en el día de hoy a lo largo de nuestro país, delitos tales como narcotráfico, que afectan a todo tipo de personas, pero más severamente nuestra juventud como también a las poblaciones que están bajo este dominio, homicidios que hasta la fecha han aumentado en comparación con el año pasado [PDI, 2022], donde la mayoría son mafias o crimen organizado realizando el famoso 'ajuste de cuentas', estafas o fraudes como trámites migratorios, robo de identidad, estafas de tipo bancarias. También están las de ventas, como por ejemplo, a la hora de hacer una compra, el producto o servicio no se realice como también el dinero que se está invirtiendo o entregando no llega a su destino o a la persona que debería, de tipo piramidal, donde la mayoría de los protagonistas de este tipo de delitos son empresarios o personas con gran habilidad y persuasión a la hora de hablar con la gente que terminan cayendo en la trampa, entre otros delitos, pero nosotros nos enfocaremos específicamente en los delitos informáticos.

Los delitos informáticos han ido en aumento en comparación con años anteriores, ya que si bien cada vez estamos entrando en una nueva era donde casi todo se realiza a través de internet, tales como las transacciones, mensajes entre personas, videollamadas por redes sociales, búsqueda de algún objeto, inmueble o servicio que requiera una persona para un determinado fin, como también la digitalización que consiste en la transformación de procesos analógicos y objetos físicos en digitales [DropBox, 2022], dirigido mayoritariamente para las empresas o pymes en crecimiento. Con todo esto se abren caminos para que personas con malas intenciones o delincuentes cibernéticos puedan romper y acceder de manera ilegal sin la autorización de algún tercero o incluso de la misma compañía o empresa a estos datos para poder buscar beneficios personales, tales como robar información personal para ser vendida a otras organizaciones y así generar más dinero, la encriptación de estos datos o archivos para poder cobrar un rescate o simplemente con la finalidad de poder hacer daño.

Otro de los tantos delitos que si bien no afecta en los recursos de una persona, sino más bien en la estabilidad mental y psicológica de una, sería el Grooming, que en sí se refiere a las situaciones de abuso sexual de niños, niñas o adolescentes por parte de adultos a través de internet, usando por parte de los acosadores herramientas de software como redes sociales o chats, y/o herramientas físicas tales como la webcam que con un software específico se puede editar la cámara tratando de hacer o poner en cámara a una persona que en realidad no es esa persona, engañando así a su objetivo.

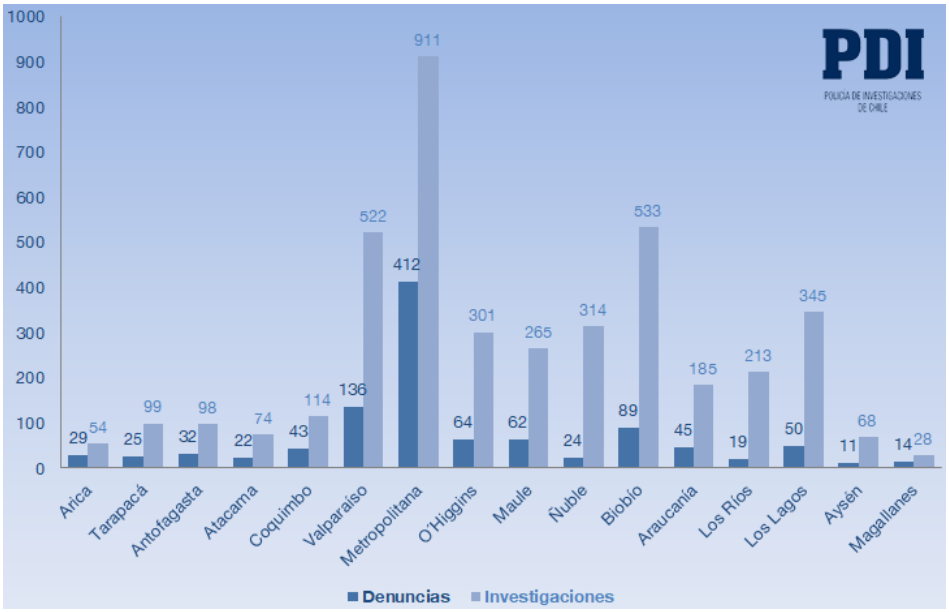


Figura 1: Gráfico de Grooming en 2019
Fuente: PDI Chile.

1.2. Organización

La Policía de Investigaciones de Chile (PDI) es una institución gubernamental de carácter civil investigativo por parte de la República de Chile, que fue creada y fundada oficialmente el 19 de junio de 1933. Compone las fuerzas del orden y seguridad de Chile, dependiente del Ministerio del Interior y Seguridad Pública, donde el personal o los miembros que componen esta institución están sometidos a un régimen jerárquico y disciplinario estricto que desarrolla labores de investigación criminalística.

Su misión fundamental es realizar investigaciones y aclaraciones policiales de los delitos, como los ya descritos al principio de esta página, y controlar el movimiento migratorio de personas del país en los pasos fronterizos, aeropuertos y puertos, representando así a nuestra nación frente a Interpol.

Sus funcionarios son llamados genéricamente detectives, que están caracterizados por no usar uniforme, contando así con una placa de servicio para la identificación de estos a la hora de realizar su trabajo. Además, llevan una casaquilla de color azul que es utilizada en momentos en que deben ser reconocidos. La labor operativa es realizada por los asistentes policiales, que, haciendo una comparación con las fuerzas armadas, se asimilan a suboficiales.

Estos detectives también desarrollan su trabajo con métodos científicos, principalmente mediante la criminalística, criminología e inteligencia policial, junto a otras disciplinas que van de la mano con la primera, como el tema balístico, dactiloscopia, medicina forense, psicolo-

gía y ciberdelitos, que será nuestro tema principal para la memoria, entre otros.

1.3. Situación Actual

Si bien para poder combatir esto nuestras organizaciones por parte del Estado ya tienen el poder y la legislación para poder combatir este tipo de delitos con la nueva ley que entró en vigor en junio del presente año (2022)[Gobierno, 2022] y que es remplazada con una que había sido creada en 1993, donde se le agregaron los siguientes delitos:

- **Acceso Ilícito:** Delito que comete quien accede a un sistema informático sin autorización o excediendo la autorización que posea [Abogados, 2022], donde este tipo de malhechos ocurren casi siempre en compañías o empresas que guardan datos de alta sensibilidad y que su mal uso puede generar pérdidas millonarias.
- **Interpretación Ilícita:** es el delito que indebidamente intercepte, interrumpa o interfiera, por medios técnicos la transmisión no pública de información en un sistema informático o entre dos o más sistemas informáticos [BCN, 2022], este otro tipo de delitos ocurren cuando hay comunicación entre dos equipos como por ejemplo el uso de WhatsApp que se envían a través de las señales electromagnéticas con la antena del dispositivo que son enviados a través de paquetes donde claramente estos paquetes pueden ser interceptados con un dispositivo especializado que tenga alguna antena o que haya sido modificado para poder cumplir tan fin y poder ver la información que posea, en el caso de que estos paquetes no estén encriptados.
- **Ataque a la Integridad de Datos Informáticos:** se le nombra así al delito que comete quien indebidamente altere, dañe o suprima datos informáticos [TI, 2022], pongámonos en un ejemplo, cuando un ciberdelincuente tiene finalmente el acceso a una base de datos de alguna empresa u organización, indiferentemente como pudo acceder a esta herramienta, este puede con los conocimientos previos de programación buscar los diferentes esquemas o tablas que puede contener la base de datos y editar con facilidad alguna tabla que contenga información como los datos personales de los diferentes usuarios que sean miembros de la compañía.
- **Falsificación Informática:** comete falsificación informática el que indebidamente introduzca, altere, dañe o suprima datos informáticos que sean tomados como auténticos o usados para generar documentos auténticos [TI, 2022], esto es un ejemplo que ocurría mucho en la época de la Pandemia del COVID-19, que al estar con restricciones de movilización, surgió un documento o pase que podía ser usado para que uno pudiera trasladarse sin ser detenido por las autoridades, todo esto solo si se tenían vacunas, pero pasó que algunas personas no tenían la vacuna o estaban convencidas de la efectividad que esta les daba contra el virus, por lo que para poder trasladarse empezó a surgir el mercado negro de estos documentos donde se les modificaba el nombre y RUT, dejando así libres y con pase a las personas que no deberían tenerlo.

- **Receptación de Datos Informáticos:** es el delito que comete quien conociendo su origen o no pudiendo menos que conocerlo, comercialice, transfiera o almacene datos informáticos mediante delitos como acceso ilícito, interceptación ilícita, y falsificación informática, este es un ejemplo que habíamos dicho en la introducción a las leyes que entraron en vigor, donde un ciberdelincuente puede vender información de la cual fue sacada de una base de datos accedida de manera ilegal y sin autorización a terceras personas tales como personas en sí (individuales) o empresas o compañías que puedan hacerle competencia a esta organización, la mayoría de estos datos.
- **Fraude Informático:** el fraude cibernético e informático se refiere al fraude realizado a través del uso de una computadora o del internet. Como habíamos hablado antes había distintos tipos de estafas donde cada una tenía un contexto diferente, que también son llevadas a la web donde páginas de marketing online como por ejemplo “mercado libre”, “yapo”, “E-bay”, entre muchas otras pueden ser usadas para cometer fraude informático, así como la venta de objetos que no corresponden con la venta original, que no tenga las características que se había dicho en la descripción de objeto o simplemente no entregar el producto, esto igual ha disminuido debido al sistema de seguridad que han impuesto este tipo de plataformas [Oliver y Mayer, 2020] como la retención del dinero por parte de las empresas hasta asegurar que el cliente tiene y está satisfecho con el producto que se le fue entregado y así finalmente pasar el dinero a la persona que le corresponde.

Todo esto hace que la institución gubernamental a la cual nosotros nos referiremos como PDI (Policía de Investigaciones) necesite llevar a cabo una investigación que a veces puede ser o no ser compleja, dependiendo del contexto y la severidad del delito que se ha cometido, para dar con el paradero del ciberdelincuente y así obtener la herramienta o dispositivo con el que se realizó el delito, que en todos los casos es un computador portátil o de escritorio, realizando así un análisis forense para obtener la evidencia y llevarlo a juicio.

Existen, a su vez, en la web o internet varias herramientas o distintos tipos de software que convierten de XML a PDF directamente, pero estos no resultan tan eficientes como se requiere o necesita, dado que muchos de ellos producen un formato de salida o plantilla no deseada para elaborar un informe impreso, es decir, salen con un formato informal y un diseño que puede ser el mismo, pero en formato PDF, lo que hace que el software no sea tan eficiente y no haya necesidad de usarlo debido a su mala traducción. Tomaremos como ejemplo un archivo XML sacado de la web y lo transformaremos a un formato PDF con una herramienta que hemos encontrado en internet:

```xml
<?xml version="1.0"?>
<Catalog>
   <Book id="bk101">
      <Author>Garghentini, Davide</Author>
      <Title>XML Developer's Guide</Title>
      <Genre>Computer</Genre>
      <Price>44.95</Price>
      <PublishDate>2000-10-01</PublishDate>
      <Description>An in-depth look at creating applications
      with XML.</Description>
   </Book>
   <Book id="bk102">
      <Author>Garcia, Debra</Author>
      <Title>Midnight Rain</Title>
      <Genre>Fantasy</Genre>
      <Price>5.95</Price>
      <PublishDate>2000-12-16</PublishDate>
      <Description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</Description>
   </Book>
</Catalog>
```

<?xml version="1.0"?>
<Catalog>
   <Book id="bk101">
      <Author>Garghentini, Davide</Author>
      <Title>XML Developer's Guide</Title>
      <Genre>Computer</Genre>
      <Price>44.95</Price>
      <PublishDate>2000-10-01</PublishDate>
      <Description>An in-depth look at creating applications
      with XML.</Description>
   </Book>
   <Book id="bk102">
      <Author>Garcia, Debra</Author>
      <Title>Midnight Rain</Title>
      <Genre>Fantasy</Genre>
      <Price>5.95</Price>
      <PublishDate>2000-12-16</PublishDate>
      <Description>A former architect battles corporate zombies,
      an evil sorceress, and her own childhood to become queen
      of the world.</Description>
   </Book>
</Catalog>

Figura 2: Ejemplo de archivo XML (izquierda) y su traducción a PDF (derecha). Fuente: Elaboracion Propia.

Como podemos apreciar, vemos que claramente la traducción de este archivo no es la adecuada, siendo inútil para un funcionario usar este archivo para temas de comunicación entre la institución o para generar evidencias a la hora de un juicio.

También existen plataformas web que ofrecen la posibilidad de transformar entre los distintos formatos de *XML* a LaTeX, como el sitio web de *ASPOSE* [ASpose, 2022], que nos permite subir un archivo. Sin embargo, al realizar la acción, se encuentran restricciones, como un límite máximo de solamente 10 archivos y la eliminación de los ya procesados después de 24 horas, lo que no garantiza un respaldo en caso de cualquier situación que ocurra. También se realizó un caso de prueba con un archivo en formato *XML* generado al azar, y como resultado se observó que el código generado por esta plataforma no era el 100 % requerido, reconociendo secciones que no debería o procesando mal la información. Todo esto se debe a que las plataformas son muy genéricas y no siguen completamente un sistema guiado de etiquetas.

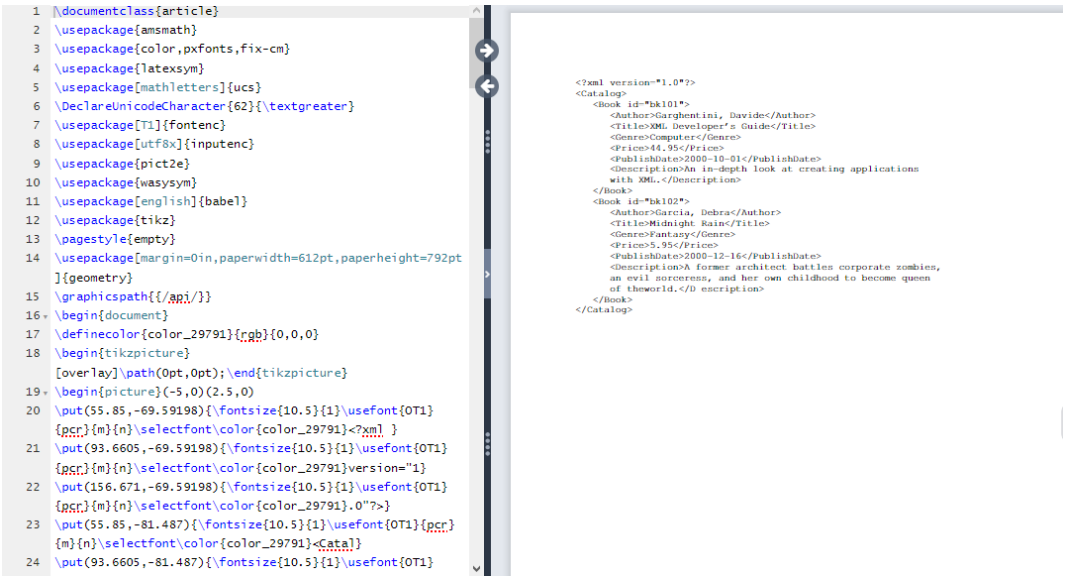


Figura 3: Ejemplo de archivo TEX transformado (izquierda) y conversión a PDF (derecha). Fuente: Elaboracion propia.

Como consecuencia de todo lo visto anteriormente con ejemplos reales hechos a través de la web, se hace necesaria la implementación de una plataforma que pueda ser 100 % personalizada por la institución y que además pueda seguir las distintas etiquetas o reglas que se generan en los informes de análisis forenses, pudiendo así seguir un protocolo preciso para evitar errores o ambigüedades en la generación de código. Claro que también se tendrá en cuenta como inspiración las distintas plataformas que ofrecen este servicio para poder adaptarlo al contexto adecuado al que se estará estudiando.

Sin embargo, se hace necesaria una solución así, dado que es un problema complejo de ingeniería, ya que estos archivos en su árbol de etiquetas que son generados no siempre tendrán el mismo formato o la misma cantidad de etiquetas, variando así con el tiempo o el tipo de dispositivo al que se está analizando. La información que tiene uno puede tener mucho más que la que tenga otro, es decir, podemos tener un archivo que pese mucho menos de 10KB o otro que tenga mucho más de 10MB, dependiendo de la información que haya sido posible extraer o la cantidad de datos que haya en la computadora portátil. Por lo tanto, en los próximos capítulos, podemos ver cómo abordar esta problemática con su solución.

1.4. Árbol del problema

A continuación, se presentará el árbol del problema que define las causas del problema que estaremos abordando a lo largo de esta investigación, junto con el problema central que este produce y las consecuencias que son asumidas por los investigadores policiales, dificultando así el trabajo profesional que realizan.

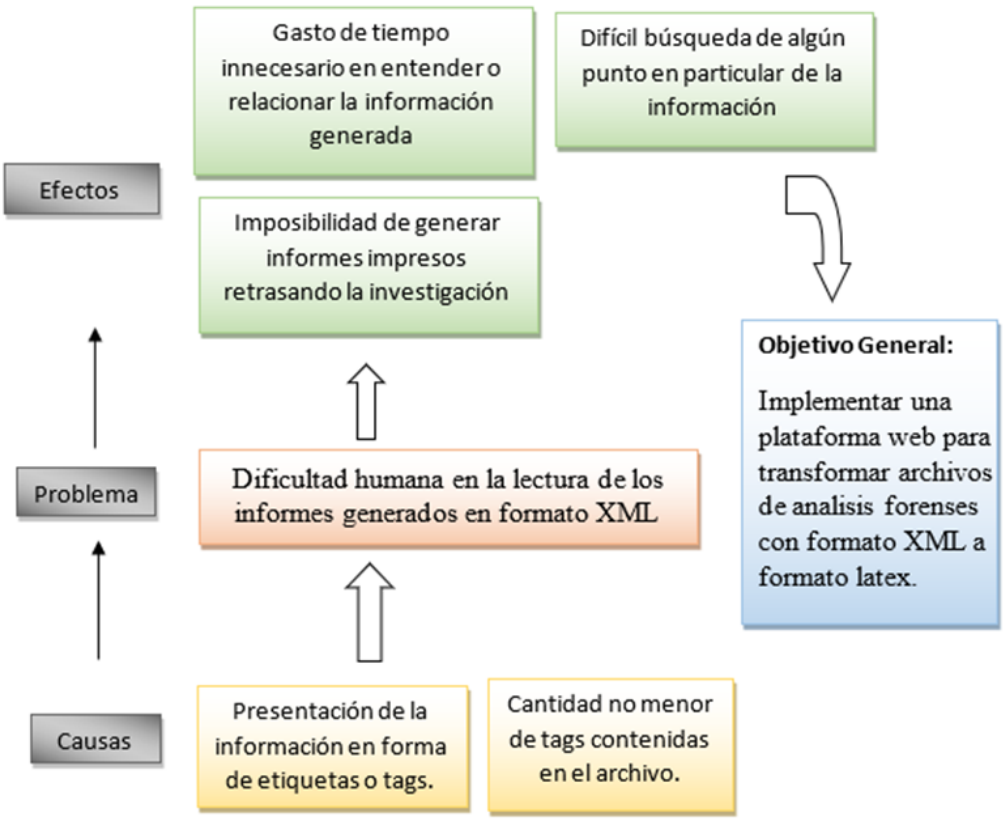


Figura 4: Árbol del problema.
Fuente: Elaboracion propia.

1.5. Objetivos

1.5.1. Objetivo General

Implementar una plataforma web para transformar archivos de análisis forenses con formato XML a formato Latex.

1.5.2. Objetivos Específicos

- Presentar y analizar el problema en la lectura de archivos generados al finalizar un análisis forense.
- Definir patrones que relacionen etiquetas a comandos específicos en Latex.
- Diseñar un algoritmo considerando los patrones definidos.

- Evaluar el prototipo considerando los diversos casos, emitidos directamente desde la institución.

1.6. Alcance

Para esta memoria, nuestro objetivo es desarrollar nuestra plataforma de tal manera que todos los archivos generados en el análisis forense, luego de la incautación o recolección de herramientas después de los delitos, puedan ser procesados convirtiendo así el formato XML con árbol de etiquetas a un archivo LaTeX. Esto facilitaría su edición y posterior transformación al formato de archivo PDF, de manera que estos sean fáciles de leer para el ojo humano, como un archivo normal. Esto cumpliría el objetivo de los funcionarios, que es poder realizar informes impresos y leerlos fácilmente.

CAPÍTULO 2
MARCO CONCEPTUAL

En este capítulo se abordará la base conceptual que dará vida al proyecto, abordando los conceptos técnicos que no son comunes en el día a día para quienes no están interiorizados en este tema, ya que ocurren en contextos muy particulares como son los delitos. También se agregarán las metodologías y herramientas con las cuales se realizará esta memoria.

2.1. Fundamentos Teóricos

2.1.1. Ánalisis forense

Una vez que la policía de investigaciones ha logrado dar con el paradero del ciberdelincuente y ha conseguido el dispositivo con el cual cometió el delito a través de la confiscación de este mismo como evidencia, se necesita realizar un análisis forense que comprenda todo un conjunto de técnicas diseñadas para extraer la información de cualquier soporte sin alterar su estado. Esto permite buscar datos ocultos, dañados o incluso eliminados, lo que puede resultar en pruebas determinantes en un proceso judicial. Este análisis comprende una serie de etapas destinadas a asegurar las evidencias encontradas, garantizando así su validez.

Para realizar un análisis forense de manera profesional, se deben seguir ciertos pasos, cada uno con su nivel de tolerancia y cuidado. Dado que existe una alta probabilidad de cometer errores debido a malas prácticas o fallos humanos, es crucial tener precaución para evitar retrasos en la investigación o incluso el fracaso de la misma, ya que los datos podrían haber sido cambiados, destruidos o suprimidos durante el análisis en profundidad [Prakmatic,]. Para ello, se requieren los siguientes conceptos:

- **Asegurar la escena:** se trata de la primera fase y no siempre se aplica; su objetivo es impedir que nadie pueda alterar algo. Esta fase se usa más que nada cuando hay aseguramiento de dispositivos usados en delitos como crimen organizado, donde hay varias personas involucradas y cualquiera de ellas tenga la intención de eliminar la evidencia, impidiendo así que los acusados por sea cual sea el delito informático cometido puedan ser juzgados, evitando así las penas de presidio.
- **Identificación de evidencias:** se trata de constatar qué evidencias deben recogerse para un análisis posterior. Se deben identificar los dispositivos y sistemas a analizar y distinguir qué evidencias hay que destacar. Todo esto se refiere a que se deben saber e identificar cuáles son los que verdaderamente estuvieron en el acto delictual, como en el caso que vamos a ver sería el computador portátil o de escritorio.

- **Adquisición de datos:** se trata de una fase crítica debido a la posibilidad de modificar por error alguna de las evidencias digitales, lo que un error de este tipo podría invalidar las pruebas en un proceso judicial. Esta es una de las fases más sensibles y que requieren mayor cuidado que hay que tener a la hora de manipular datos, dado que la falta de experiencia o conocimientos al manejar datos podría resultar en que la evidencia no sea válida, dejando así libres a los imputados.
- **Análisis de datos:** Es la etapa en la que se busca información útil y necesaria en relación con las evidencias del caso o delito que fue hecho; en este punto se estudian los ficheros donde puede estar la información eliminada, registros, logs del sistema, ficheros, etc. Todo esto será importante a la hora de llegar a un juicio contra una persona, dado que es la evidencia que podría llevarla al presidio.
- **Presentación de informes o resultados:** Finalizado el análisis forense, se debe redactar un informe pericial con conclusiones y justificación del sistema de trabajo empleado. El informe debe ser claro y conciso, ya que puede terminar como prueba en los tribunales.

Todo este proceso que realiza la policía de investigaciones del tratamiento o búsqueda de datos en dispositivos se genera finalmente en un archivo en formato XML. Si bien tenemos la información ya extraída, es aquí donde se enfoca el problema y la justificación de esta memoria, dado que los datos recolectados aparecen de forma ilegible para el ojo humano, lo que dificulta que los funcionarios redacten informes impresos para temas formales y la comunicación entre ellos mismos. Para resolver esto, es necesario conocer la definición de un archivo en formato XML.

2.1.2. XML

XML comenzó a desarrollarse en 1996 auspiciado por el W3C (World Wide Web Consortium), una institución que genera recomendaciones y estándares que aseguran el crecimiento de la World Wide Web a lo largo del tiempo. Este lenguaje fue desarrollado con un claro propósito: diseñar un lenguaje de marcado optimizado para internet. Con ”lenguaje de marcado”nos referimos a una forma de codificar un documento que incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación. XML es la base de HTML, el lenguaje más extendido [Amazon, 2022] y usado ampliamente en la web, este tipo de lenguaje también sirve para la comunicación entre sistemas computacionales debido a su descripción autocontenido. Un claro ejemplo del uso de este lenguaje es en los sistemas distribuidos, donde la comunicación entre sistemas es el tema principal. Entre los otros grandes beneficios que posee, podemos encontrar:

- **Respaldo para las transacciones interempresariales**: Cuando una o varias empresas venden bienes o servicios a otras empresas, ya sean pequeñas, medianas o grandes,

estas dos empresas necesitan intercambiar información como los costos del bien que se está ofreciendo, especificaciones de cualquier tipo como descripciones, notas adicionales o incluso reglas de negocio poniendo límites a la otra entidad, compartiendo la información necesaria de manera electrónica y cerrando negocios complejos de forma automática, sin intervención humana.

- **Conservación de la integridad de los datos**: Este tipo de archivos, al transferir los datos junto con la descripción de los datos, que son llamados dentro de la comunidad que ocupa estos archivos como **tags** o **etiquetas**, evita la pérdida de la integridad de los datos, pudiendo así verificar la precisión de estos, personalizar automáticamente la presentación de datos para diferentes usuarios o almacenar datos de forma coherente en múltiples plataformas.
- **Mejora de la eficiencia de búsqueda**: Google, conocido como el motor de búsqueda más grande en el mundo, así como también lo son duckduckgo, bing, Yahoo!, entre otros, pueden ordenar y categorizar archivos XML de forma más eficiente y precisa que otros tipos de documentos. Si ponemos el ejemplo de la búsqueda de un auto con la palabra Marca, esta puede ser un sustantivo o un verbo, pero si nos basamos en las etiquetas XML, los motores de búsqueda pueden categorizar con precisión ”marca”para resultados de búsqueda relevantes. Por lo tanto, ayuda a interpretar el lenguaje natural de manera más eficiente.

XML es un lenguaje base para muchos otros, tales como HTML. HTML es un lenguaje de marcado que consta de diversas etiquetas XML, las que posteriormente son interpretadas por los navegadores web para mostrar la información de manera ordenada y legible. Se usa en la web de manera omnipresente, ya que es el lenguaje que se utiliza para estructurar y presentar contenido en la web [Holcome, 2022]. A continuación, se presentan ejemplos de archivos en formato HTML y XML.

```
<HTML>
<HEAD>
<TITLE>Ejemplo01.htm</TITLE>

<SCRIPT LANGUAGE="JavaScript">
  //Visualizar un mensaje de bienvenida
  alert("¡Bienvenido a nuestra página!");
</SCRIPT>

</HEAD>
<BODY>
<a href='Ejemplo02.html'>Ir al siguiente ejemplo...</a>
</BODY>
</HTML>
```

Figura 5: Ejemplo de archivo HTML.
Fuente: Elaboración propia.

En este ejemplo podemos ver un HTML normal con un link de referencia a otra página.

```
1  <?xml version="1.0"?>
2  <Book>
3      <Author>
4          <name>ALberto hurtado</name>
5          <age>16</age>
6          <childrens>5</childrens>
7      </Author>
8      <Title>XML Developer's Guide</Title>
9      <Genre>Computer</Genre>
10     <Price>44.95</Price>
11     <PublishDate>2000-10-01</PublishDate>
12     <Description>An in-depth look at creating
           applications with XML.</Description>
13 </Book>
```

Figura 6: Ejemplo de archivo XML.
Fuente: Elaboración propia.

Con este ejemplo de archivo en formato XML podemos ver claramente la diferencia con un HTML. Para ser más específicos, tenemos un libro que estará definido con la etiqueta *Book*, y que dentro de esta tendrá las características más importantes que definen el escrito, como el autor, título, género, precio, fecha de publicación y descripción.

Ahora se puede observar también que dentro de la etiqueta autor, tenemos aún más etiquetas que describen las otras características propiamente tales del autor, como su nombre, edad y en este caso la cantidad de hijos que tiene.

2.1.3. PDF

PDF, que por sus siglas en inglés *Portable Document Format*, es un formato de almacenamiento de documentos digitales independientemente de las plataformas de software o hardware que se estén utilizando, ampliamente utilizado en todo el mundo oficialmente desde el 1 de julio de 2008, publicado por la Organización Internacional de Estandarización como *ISO 32000-1*. Pero, ¿cuáles son las características de este formato que lo hacen tan especial? A continuación se presenta un listado de las más importantes por su uso.

- Un documento PDF tiene la misma apariencia, color, tipo de imprenta, gráficos y a la vez el mismo formato que un documento impreso. Esto permite que puedan ser digitalizados y almacenados en una computadora para su posterior utilización en asuntos formales, como envío de información entre personas y/o entidades, incluyendo su eventual impresión para obtener una copia física del documento. [Anjelino, 2022]
- Las características anteriores ayudan enormemente a su distribución por la Web, ya sea mediante correos electrónicos, dispositivos *USB* o incluso teléfonos celulares, donde es muy utilizado para compartir información gráfica o textual, como manuales, contratos, entre otros documentos.
- En temas de seguridad, los archivos en formato *PDF* pueden brindar varias alternativas, debido a que no se pueden modificar fácilmente. Ofrece opciones para ajustar niveles de acceso para proteger el contenido y todo el documento, como filigranas, contraseñas o firmas digitales. Los bancos y empresas que trabajan con datos delicados son los que más utilizan estas medidas de seguridad. [Adobe, 2022]

A continuación se presenta una imagen que contiene un ejemplo de un documento en formato PDF, donde se observa que es solamente de lectura, ya que los archivos con este formato no fueron creados para ser editados, con el fin de evitar problemas de integridad de los datos generados en *PDF*.

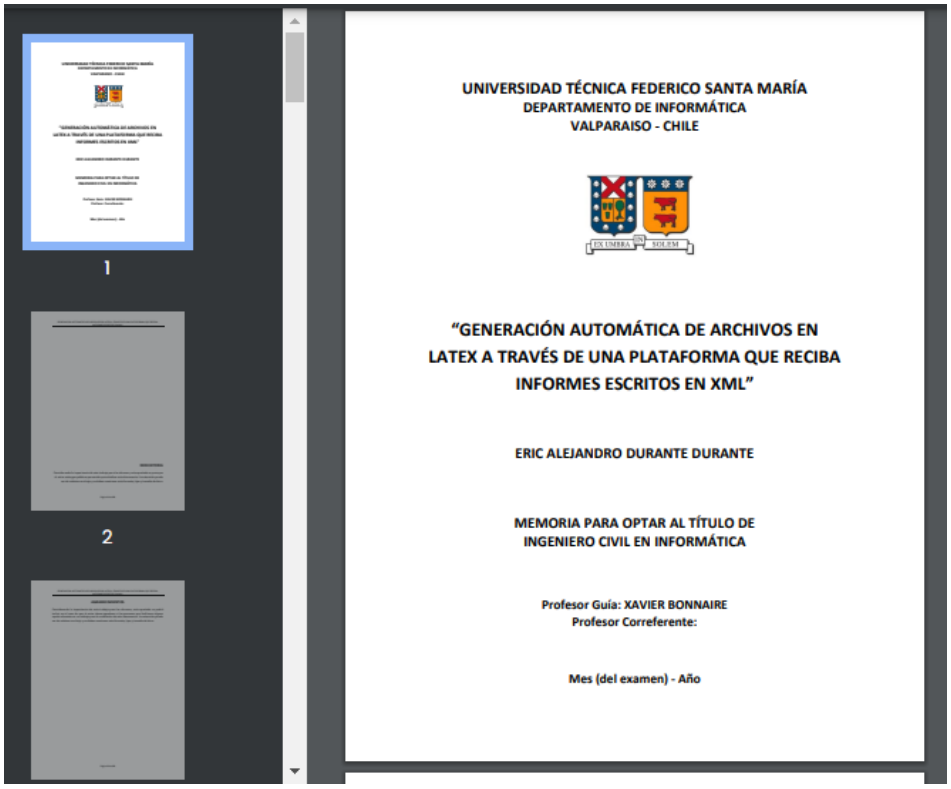


Figura 7: Ejemplo de un buen archivo PDF.
Fuente: Elaboración propia.

En la siguiente imagen se muestra el mismo tipo de archivo, solicitando contraseña, algo común cuando existen datos de índole privada y solo ciertos individuos necesitan tener acceso a la información presentada en el documento.

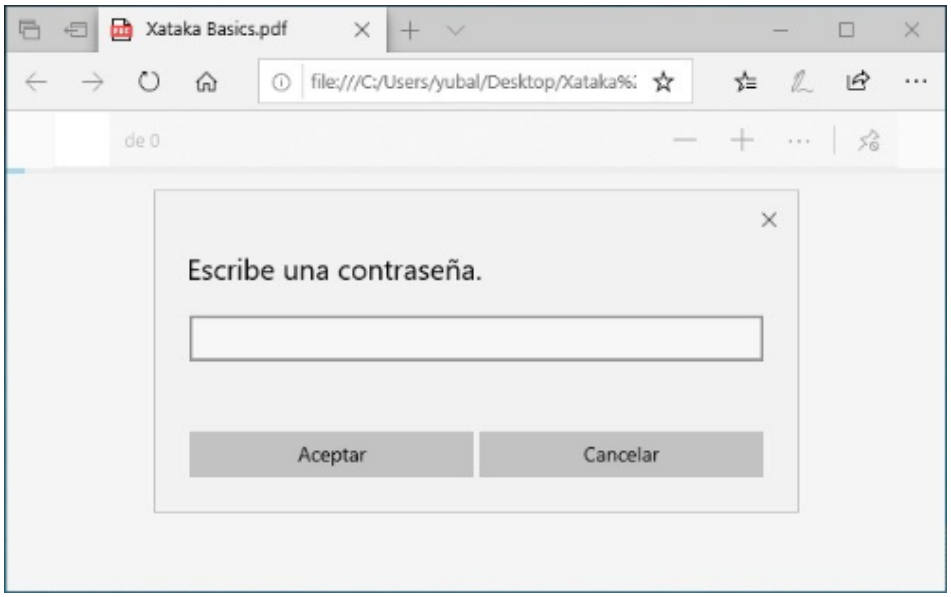


Figura 8: Ejemplo de un buen archivo PDF.
Fuente: Xataka.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

La siguiente propuesta de solución se realizó en base a una medida general que puede ser escalable y mantenible en el tiempo. Esto quiere decir que estaría resolviendo el problema descrito y explicado anteriormente, pero que de alguna forma u otra puede llegar a solucionar problemas relacionados con archivos en formato *XML* que necesiten ser traducidos a formato LaTeX, ya sea para mejorar su legibilidad o para facilitar la edición. Al trabajar con etiquetas, podemos cometer el error de editar nuestro contenido con la estructura de etiquetas, lo que puede afectar el diseño esperado.

3.1. Metodología de trabajo

La metodología de trabajo que emplearemos se determinará según el contexto y la simplicidad del desarrollo que abordaremos en este capítulo. Dado que se trata de un solo desarrollador, autor de este documento y creador de la plataforma, optaremos por una metodología que se base en el entorno en el que estamos trabajando.

Una de las metodologías consideradas es **Kanban**, la cual es un método para gestionar el flujo de trabajo, ampliamente reconocido en el desarrollo de software y la fabricación, y aplicable a una amplia gama de campos. Originado en Toyota entre los años 40 y 50 como parte de su sistema de producción Just-in-Time.

La palabra ”Kanban”proviene del japonés y significa literalmente ”tarjeta visual.º ”tabla visual”. En su forma más básica, Kanban utiliza tarjetas o notas para representar unidades de trabajo. Estas tarjetas se desplazan a través de un tablero que muestra diferentes etapas del proceso, como ”Por hacer”, ”Desarrollo” y ”Terminado”. Los equipos pueden ver fácilmente qué trabajo está en marcha, qué se ha completado y qué queda por hacer.

Kanban es una metodología flexible que se centra en la mejora continua y es comúnmente utilizada por equipos. Lo más destacable es que también puede ser adaptada para uso personal. Un desarrollador individual puede mantener un tablero Kanban para visualizar y priorizar su propio trabajo, lo que facilita la gestión de tareas y la concentración en lo que es importante en cada momento, lo que resulta muy adecuado en el contexto en el que nos encontramos.

Otras metodologías como **Scrum** o **Lean** son mucho más adaptables para equipos u organizaciones con más personal, dado que tienen una serie de eventos como las reuniones diarias de **Scrum**, donde el equipo se reúne cada día para revisar su progreso en las tareas designadas, o la retrospectiva de **Scrum**, donde el equipo se reúne al final de cada sprint (proceso de desarrollo del software) para discutir temas sobre qué mejorar y qué dejar de hacer. Es-

to, según la definición, es recomendable cuando se trabaja con más de una persona en un proyecto debido al objetivo que tiene cada evento.

Es por esto que utilizaremos la metodología **Kanban** para el trabajo personal. A continuación se detallan aspectos claves de la misma:

- **Tablero Kanban:** El tablero Kanban es el elemento central de esta metodología, dividiéndose en columnas que representan diferentes etapas del flujo de trabajo. Para este proyecto tendremos las columnas ”Por hacer”, ”Desarrollo” y ”Finalizado”. Cada tarea se mueve a través del tablero desde la columna ”Por hacer”hasta ”Hecho.ª medida que avanza en el proceso.
- **Visualización del flujo de trabajo:** Kanban se enfoca en visualizar el flujo de trabajo de principio a fin, permitiendo una comprensión clara del estado de las tareas y dónde se encuentran en el proceso en cualquier momento.
- **Gestión de cambios:** Kanban es flexible y permite la gestión ágil de cambios en las prioridades o requisitos del proyecto. Las tareas pueden agregarse, eliminarse o reorganizarse fácilmente en el tablero Kanban según sea necesario para adaptarse a las necesidades del proyecto.

Kanban no tiene fases definidas como otras metodologías, pero en este proyecto nos centraremos en la visualización y optimización del flujo de trabajo a través de un enfoque flexible y adaptable que se adapte a los cambios que mejoren el desarrollo del mismo.

3.2. Antecedentes

Como pudimos observar anteriormente, existían sitios web que ofrecían los servicios de traducción para diferentes archivos. En este contexto particular al que nos referimos, nos enfocaremos en el archivo *XML*. Si bien podríamos desarrollar un software que se instale localmente para la institución de la PDI, esto sería muy engorroso debido a los diferentes puntos que serán explicados:

- **Ordenador de gran potencia**: Para poder tener un software que funcione de la mejor calidad posible, necesitaremos un computador con buenos requisitos [SonTips, 2024]. Esto implica tener una memoria RAM superior a los **4 GB**, lo que significa que se podrían tener más programas abiertos de los que ya se tienen; una tarjeta **NVIDIA** o **AMD** para realizar las operaciones gráficas del computador, es decir, programas más fluidos y mejores gráficos; unidades de disco sólido **SSD**, que en comparación con los **HDD**, trabajan mucho más rápido a la hora de realizar procesos; y, sobre todo, un buen procesador, que es el cerebro del ordenador y se encarga de procesar la información y leer las instrucciones entrantes.

- **Actualizaciones**: Cuando tenemos un software, debemos mantenerlo actualizado en todos los ordenadores con la última versión. Ya sea por razones de seguridad, nuevas funcionalidades o errores que puedan tener los programas, es importante tenerlos en su estado más reciente en todos los computadores en los que estén instalados, especialmente en una institución que realiza labores para el bien de la sociedad.
- **Fallas**: Las fallas que puede tener un software de manera local pueden ser numerosas, ya sea por errores que comete el mismo usuario, incompatibilidades con el sistema operativo en el que está instalado o incluso problemas inherentes al propio software. Resolver estos problemas puede llevar tiempo adicional, y pueden ser evitados si se utilizan otros métodos.

Con todos estos antecedentes presentados, podemos destacar la sugerencia del profesor Xavier Bonnaire (Profesor guía), quien en una reunión realizada el miércoles 13 de abril, propuso la opción de desarrollar una plataforma web que pudiera cargar los archivos y realizar las respectivas traducciones.

3.3. Modelo de la solución

Con los antecedentes mencionados anteriormente, se propone un modelo de evaluación para la traducción de código *XML* a LaTeX, esquematizado en la siguiente figura:

Figura 9: Modelo de la solución
Fuente: Elaboración propia

- **Archivos necesarios:** Los archivos necesarios corresponden a todos los documentos necesarios para poder realizar la traducción completa al formato deseado. En este caso, los archivos son:
  - El archivo en formato *XML* que se desea traducir. Este archivo es el que la plataforma destinará para su objetivo y sus funciones, dado que es el más importante

y el que finalmente será transformado al código en formato LaTeX para luego convertirse a formato PDF.

- Un archivo en texto plano que contenga cada una de las traducciones de los tags o etiquetas *XML*. Es decir, cada etiqueta del archivo en formato *XML* será transformada al código LaTeX que se generará, permitiendo así que la plataforma realice los cambios necesarios.
- Un archivo en formato *TEX*, que será la plantilla o, en otras palabras, la portada y el final del documento que tendrá el archivo LaTeX final que se traducirá. Esto es una ventaja en el ámbito de que si se quiere utilizar el archivo traducido para un informe impreso en el contexto de empresas o instituciones importantes, como lo es la Policía de Investigaciones, puede ayudar a la legibilidad y la formalidad del documento.

- **Transformación de datos:** Otra parte importante de la plataforma web es la transformación de datos. Esta parte toma los datos del archivo en texto plano o que contiene la transformación de los diferentes tags o etiquetas del *XML* al formato LaTeX, y los almacena en un diccionario para poder ser utilizados al leer el *XML*. Luego de eso, se leerá el archivo *XML* cargado, recorriendo las diferentes etiquetas que este contenga y siendo traducidas en ese mismo momento, utilizando el diccionario mencionado anteriormente. Finalmente, se limpiará el archivo de todas las etiquetas que no hayan sido traducidas o que no hayan estado en el archivo de transformaciones, para evitar errores al leer o convertir el documento en formato LaTeX descargado a *PDF*.

- **Archivos procesados:** Los documentos procesados corresponden a los archivos que fueron finalmente traducidos en el punto anterior. Al hablar de documentos, nos referimos al archivo LaTeX que fue generado por la plataforma y al archivo *PDF* final que fue convertido en el mismo instante. Cabe destacar que se procederá a la descarga del archivo *TEX* para cualquier edición posterior del documento.

Este tipo de modelo puede estar sujeto a cambios debido a que, en primera instancia, desarrollamos una plataforma orientada a la flexibilidad y la eficiencia de uso para el usuario. Al tener la posibilidad de subir un archivo con las etiquetas *XML* y sus transformaciones, ofrecemos al usuario la capacidad de transformar de manera subjetiva todos los tags del *XML* que desee, pudiendo incluso eliminar automáticamente una etiqueta si así lo desea.

También queremos destacar que al hablar de eficiencia de uso nos referimos a que nuestro usuario pueda utilizar y comprender la herramienta que estamos desarrollando para su disposición de la manera más simple posible. Es decir, al poder utilizar la plataforma, el usuario pueda hacerlo de la manera más sencilla, sin tener que recurrir a funciones o páginas adicionales tediosas que puedan retrasar el objetivo, e incluso que el propio usuario pueda aprender fácilmente sin necesidad de leer la documentación del sitio web.

Figura 10: Página principal de la plataforma
Fuente: Elaboración propia

La imagen mostrada anteriormente es la página principal de la plataforma donde entrará en funcionamiento el software creado que se denominará **Lifyx**, una combinación de las letras de LaTeX, de donde proviene la **L**, un sufijo **ify** que se utiliza a menudo para indicar la acción de convertir o transformar algo, y **XML**, que es el formato que va a convertir o transformar la plataforma.

3.4. Tecnología usada

La tecnología usada para esta plataforma debe ser elegida de acuerdo a las características o el ambiente en donde estemos analizando e implementando una solución para este problema. No podremos utilizar un script bash para ejecutar una línea de comando de manera local, siendo que se estará implementando una plataforma, o ocupar microservicios en una plataforma en la nube, siendo que el problema analizado es acotado de acuerdo a las características o el producto final que se desea lograr.

Existen varias plataformas como Wix o Soft que permiten diseñar, personalizar e incluso crear el propio código en el mismo sitio web para empezar el emprendimiento de una empresa, la cual queda enlazada con el servicio que se va a brindar. Sin embargo, el problema de estas aplicaciones es que al ser **low code**, se enfocan en la creación de aplicaciones utilizando herramientas visuales y de arrastrar y soltar en lugar de escribir código manualmente.

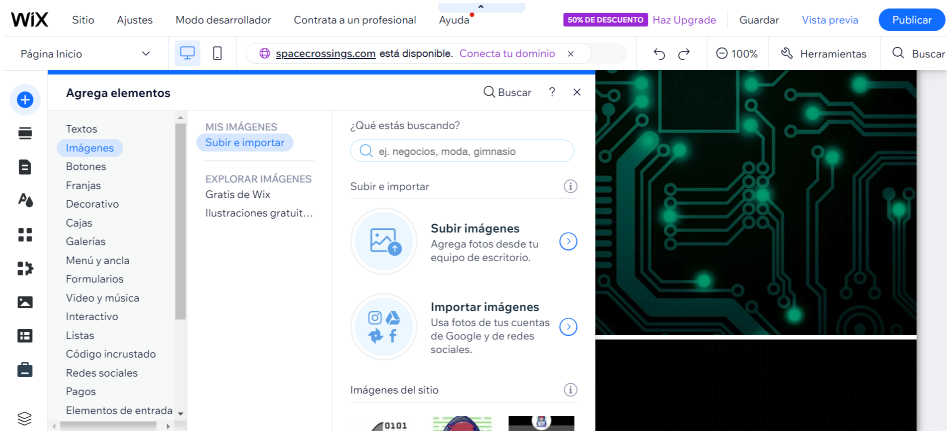


Figura 11: Ejemplo desarrollo en Wix
Fuente: Elaboración propia

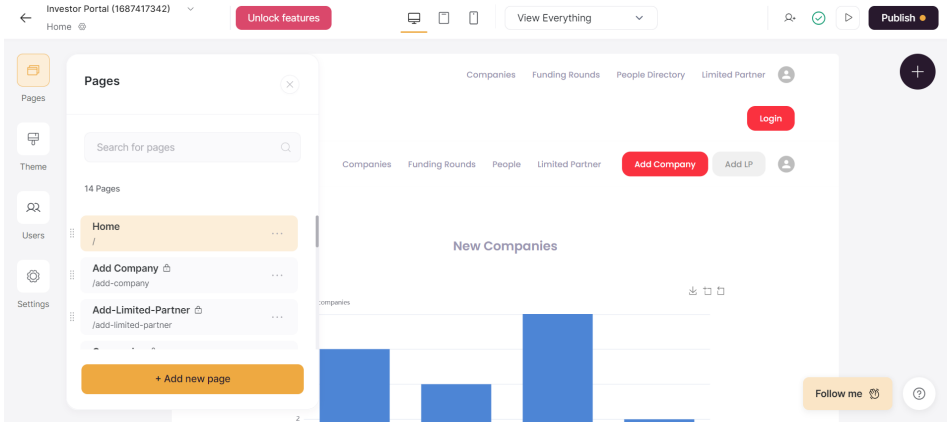


Figura 12: Ejemplo desarrollo en softr
Fuente: Elaboración propia

Como se puede observar en las dos imágenes presentadas para cada plataforma, se muestra el desarrollo que se puede hacer en las plataformas antes descritas. Sin embargo, al ser plataformas de bajo código, tienen la inflexibilidad de poder personalizar el desarrollo o el proceso que podría mejorar la solución propuesta ante el problema que queremos resolver en este informe. Por lo tanto, nuestra opción de desarrollarla en sitios web como estos queda descartada y deberá utilizarse una herramienta que sea 100 % flexible, dependiendo de las necesidades que tenga nuestro usuario, de manera que pueda ser escalable para futuros problemas que puedan presentarse incluso después de haber terminado esta investigación. Esto nos lleva a utilizar los ***frameworks*** o estructuras de código desarrolladas en algún lenguaje para satisfacer esas necesidades.

Otra necesidad que debemos agregar a la plataforma es el uso de un historial o registro

de acontecimientos que servirá en el futuro para analizar los distintos eventos que vayan ocurriendo durante el uso del software. Es decir, poder implementar un sistema de logs, ya sea en un simple sistema de archivos o en una base de datos, para la grabación de los procesos en particular. Los logs pueden ser esenciales tanto desde el punto de vista de la seguridad como para el análisis del uso del sistema, el rendimiento o la detección de fallos.

3.5. Frameworks

Dado que teníamos opciones para desarrollar nuestra solución, pero que fueron descartadas por las diferentes limitaciones que estas traían, se ha decidido elegir un **framework**. Pero para comenzar, ¿qué es un framework?

Un Framework es un conjunto de herramientas, bibliotecas y componentes predefinidos que proporcionan una estructura y funcionalidades comunes para el desarrollo de aplicaciones [Unir, 2022]. Un framework establece una base sobre la cual los desarrolladores pueden construir sus aplicaciones, brindando una serie de características y funcionalidades listas para usar.

Actúa como una plataforma de trabajo que ayuda a los desarrolladores a escribir código más eficiente y de manera más rápida, al proporcionar una estructura predefinida que define cómo deben organizarse y relacionarse los diferentes componentes de una aplicación. Esto permite a los desarrolladores centrarse en la lógica específica de su aplicación en lugar de tener que ocuparse de la infraestructura básica y las tareas repetitivas.

Pueden ser utilizados en una amplia variedad de aplicaciones, incluyendo desarrollo web, desarrollo de aplicaciones móviles, desarrollo de juegos y más. Ejemplos populares de frameworks [Frisoli, 2023] incluyen **Django**, desarrollado en base al lenguaje Python; **Gin Gonic**, basado en el lenguaje de programación Go; **Spring Boot**, desarrollado en base al lenguaje de programación Java; **NodeJS**, que utiliza JavaScript; **Flutter**, para desarrollo de aplicaciones móviles; **Unity**, para desarrollo de juegos; y la última opción, **Ruby on Rails**, hecho en base al lenguaje Ruby para el desarrollo web.

Para hacer un filtro más adecuado, sabemos que la plataforma que debemos hacer no está orientada a los videojuegos, por lo tanto, se descartaría el Framework de Unity. También tenemos que considerar que nuestra aplicación debe ser orientada a una aplicación de escritorio y no para un dispositivo móvil, lo que significa que Flutter también estaría descartado. De todos modos, puede ser usado posteriormente solo como una opción, pero no como la principal solución.

Debido a lo anterior, debemos tener en cuenta que nuestra aplicación debe tener una interfaz en la que el usuario pueda interactuar con ella, un controlador para realizar y procesar las peticiones, además de una base de datos para almacenar información adicional que será explicada en la siguiente sección. Por lo tanto, debemos analizar cada uno profundamente y

ver cuál es el más adecuado a nuestro contexto.

3.5.1. NodeJS

Node.js es un entorno de ejecución de código abierto que está construido sobre el motor JavaScript V8 de Google Chrome. Permite ejecutar código JavaScript tanto en el lado del servidor como en el lado del cliente. A diferencia de JavaScript en el navegador, que se ejecuta en el entorno del navegador, Node.js se ejecuta fuera del navegador y proporciona características adicionales y capacidades para la ejecución de aplicaciones en el servidor.

Node.js utiliza un modelo de entrada y salida sin bloqueo y orientado a eventos, lo que significa que es altamente eficiente y puede manejar una gran cantidad de solicitudes simultáneas sin bloquear el hilo de ejecución principal. Esto lo hace especialmente adecuado para aplicaciones en tiempo real y basadas en eventos, como aplicaciones web, servidores de chat, aplicaciones de transmisión en tiempo real, servicios web y mucho más.

Una de las características clave de Node.js es su capacidad para trabajar con módulos y paquetes. Utiliza el sistema de módulos CommonJS, que permite a los desarrolladores dividir su código en módulos reutilizables y gestionar las dependencias entre ellos. Node Package Manager (NPM) es el administrador de paquetes de Node.js y proporciona acceso a miles de módulos y bibliotecas preexistentes que pueden ser utilizados en las aplicaciones.

Node.js ha ganado una gran popularidad en los últimos años en todo el mundo debido a su eficiencia, escalabilidad y su gran ecosistema de módulos y bibliotecas. Es ampliamente utilizado por desarrolladores web, llegando a abarcar un 50 % de todos los proyectos que existen [Vailshery, 2023]. Ha sido adoptado por numerosas empresas para construir aplicaciones web y servicios en tiempo real, como por ejemplo empresas orientadas al ámbito laboral como **LinkedIn**, de transporte como **Uber**, de pago a nivel internacional como lo es **PayPal**, o incluso empresas de streaming como **Netflix** [Eseme, 2023].

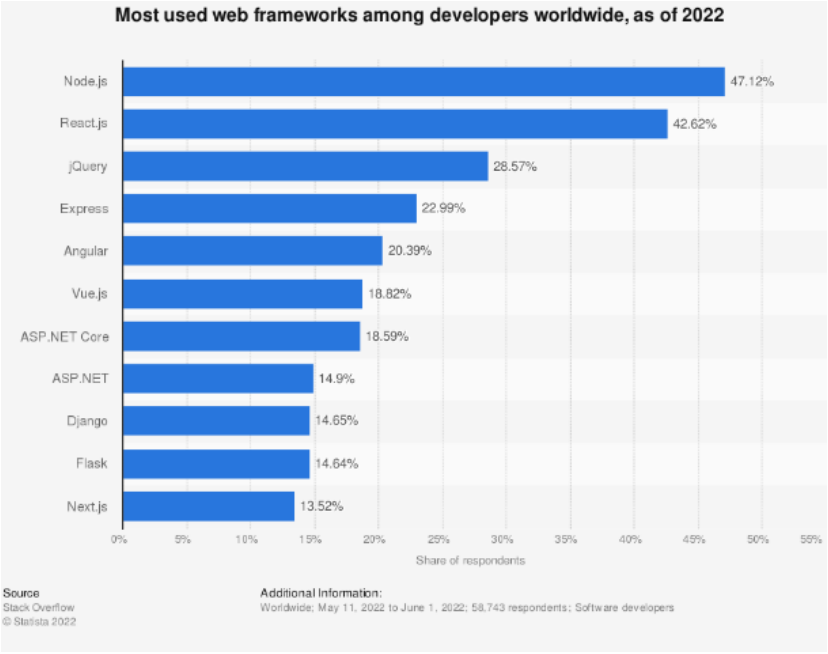


Figura 13: Popularidad de NodeJS
Fuente: Pixeladas.com

Como bien se dijo anteriormente, Node.js utiliza JavaScript como lenguaje de programación. JavaScript es un lenguaje de programación interpretado, orientado a objetos y de alto nivel. Fue creado originalmente para agregar interactividad a las páginas web y se ha convertido en uno de los lenguajes más utilizados en el desarrollo web. JavaScript se ejecuta en el lado del cliente, es decir, en el navegador web de un usuario, y se utiliza para manipular y controlar elementos de una página web, realizar validaciones de formularios, interactuar con APIs, realizar peticiones HTTP, crear animaciones, entre muchas otras cosas.

A lo largo de los años, JavaScript ha evolucionado y se ha expandido más allá del entorno del navegador. Ahora también se puede utilizar en el lado del servidor con Node.js, como se mencionó anteriormente. Esto significa que JavaScript puede ser utilizado para crear aplicaciones web completas, tanto en el cliente como en el servidor, lo que permite compartir código y lógica entre ambos lados. Es muy versátil y flexible, además de tener las siguientes características [Web, 2023]:

- Es un lenguaje **Imperativo y estructurado**, lo que significa que las sentencias se ejecutan de manera secuencial y se basa en el uso de estructuras de control bien definidas para construir el programa, centrándose en la organización lógica y ordenada del código.
- En JavaScript no se define el tipo de una variable al instanciarla. El tipo de la variable se le asignará atendiendo al valor que le asignemos, lo que hace que sea de tipado **débil**. Además, si este valor va cambiando, también lo hará el tipo de dato de la variable, haciéndola también de **tipado dinámico**.

- Es un lenguaje **Interpretado**, lo que significa que no realiza ningún proceso de compilación a código de máquina, sino que necesita de un intérprete para poder obtenerlo y ejecutar sus instrucciones. Para mejorar su uso existen motores como Chrome V8, que utiliza un compilador JIT, definido como un entorno de ejecución que mejora el rendimiento de aplicaciones compilando códigos de bytes en código de máquina nativo en tiempo de ejecución [IBM, 2023].
- Tiene Programación orientada a objetos (POO), donde el diseño de software se organiza alrededor de datos u objetos en vez de funciones y lógica.
- Al ser un lenguaje que puede ser utilizado tanto en el cliente como en el servidor, le permite tener múltiples intérpretes en diversos entornos, presentándose en diferentes sistemas operativos como *Windows*, *Mac* y *Linux*, haciéndose un lenguaje **Multiplataforma**.

Volviendo a Node.js, ¿qué ventajas y desventajas tiene este entorno de ejecución? Esta es una pregunta importante que se debe realizar, ya que se verá si esta herramienta puede adaptarse a las necesidades que nosotros requerimos.

Tabla 1: Ventajas y desventajas de Node.js
Fuente: OpenInova

Ventajas	Desventajas
Su arquitectura basada en eventos y sin bloqueo permite manejar varias conexiones concurrentes sin agotar los recursos del sistema.	Dificultades con tareas que requieren alto uso de CPU. Debido a su naturaleza de un solo hilo, las operaciones que bloquean la CPU afectan su rendimiento general.
Utiliza JavaScript tanto en el lado del servidor como en el cliente, lo que permite utilizar las mismas habilidades y bibliotecas en ambos lados.	La programación asíncrona puede ser más compleja que la programación secuencial, ya que las devoluciones de llamada pueden generar código anidado y difícil de mantener, requiriendo el uso de promesas y **async/await**.
Su comunidad activa y un ecosistema de módulos y paquetes amplio. Existen miles de módulos disponibles a través de NPM (*Node Package Manager*), lo que facilita la reutilización de código y acelera el desarrollo de aplicaciones.	Experimenta un crecimiento rápido, algunas bibliotecas y módulos pueden no tener la misma madurez y estabilidad que los disponibles en otros lenguajes o entornos, requiriendo una selección cuidadosa de las bibliotecas.
Es eficiente en la manipulación de flujos de datos, lo que lo hace ideal para aplicaciones en tiempo real que requieren transmisión de datos continua.	Limitaciones en la escalabilidad vertical, es decir, en la capacidad de aprovechar los núcleos de CPU de manera eficiente, requiriendo clústeres o procesos hijos para aprovechar los recursos del sistema.

Si bien tiene una comunidad activa y un ecosistema de módulos y paquetes amplio, lo que facilita la reutilización de código y acelera el desarrollo de aplicaciones, también tiene la desventaja de que algunas bibliotecas y módulos pueden no tener la misma madurez y estabilidad que los disponibles en otros lenguajes o entornos, requiriendo una selección cuidadosa de las bibliotecas.

Por último, resolveremos nuestra última incógnita que gira en torno al framework que estamos analizando: ¿Por qué Node.js es tan popular? Si bien las características de rapidez y eficiencia parecieran ser razones suficientes para poder responder esta pregunta, la verdad es que no lo son.

Node.js no es un framework, es simplemente un entorno de ejecución para JavaScript. Esto significa que aún debemos elegir un framework que se adapte a nuestras necesidades. Justamente este es el punto más débil de Node.js [Chakray, 2022], ya que no tiene un framework

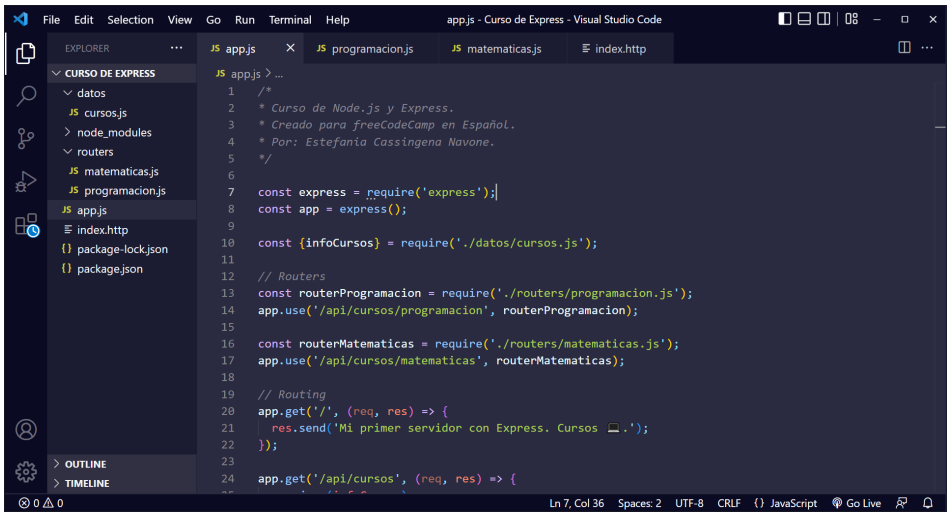


Figura 14: Programando en NodeJS
Fuente: freeCodeCamp

que sea tan popular como Django, Spring Boot o Ruby on Rails. Por lo tanto, se descarta esta opción y se procede a analizar el siguiente framework.

3.5.2. Spring Boot

Java Spring Boot es un framework de desarrollo de aplicaciones web y servicios REST basado en Java y la plataforma Spring. Spring Boot proporciona un enfoque simplificado y rápido para desarrollar aplicaciones Java, eliminando gran parte de la configuración y la complejidad asociadas con la creación de aplicaciones basadas en Spring.

Spring Boot se basa en el marco de trabajo Spring, que es uno de los más populares para el desarrollo de aplicaciones empresariales en Java [Huet, 2023]. Proporciona un conjunto de bibliotecas y herramientas que facilitan el desarrollo, la configuración y la implementación de aplicaciones Java.

La principal característica de Spring Boot es su capacidad para crear aplicaciones autocontenidas que se pueden ejecutar como JAR (Java Archive) o WAR (Web Archive) sin necesidad de un servidor de aplicaciones externo. Spring Boot también incluye un servidor integrado basado en Tomcat, Jetty o Undertow, lo que facilita aún más la creación y ejecución de aplicaciones web.

Además de simplificar la configuración, Spring Boot también proporciona una serie de características y extensiones que facilitan el desarrollo, como la gestión automática de dependencias, la configuración automática basada en convenciones y la capacidad de crear fácilmente servicios RESTful.

Este framework está basado en el lenguaje de programación Java, que es un lenguaje de programación de alto nivel y orientado a objetos. Fue creado por James Gosling y su equipo en Sun Microsystems (ahora propiedad de Oracle) y se lanzó oficialmente en 1995.

Figura 15: Ejemplo de suma en código Java
Fuente: Elaboración propia

Desde entonces, Java se ha convertido en uno de los lenguajes de programación más populares y ampliamente utilizados en el mundo del desarrollo de software [CampusMVP, 2023], y posee las siguientes características:

- Java es un lenguaje **portátil**, lo que significa que los programas escritos en Java pueden ejecutarse en diferentes plataformas sin necesidad de realizar cambios en el código fuente. Esto se logra mediante la compilación de programas Java en bytecode, que se ejecuta en una máquina virtual Java (JVM) específica para cada plataforma.
- Es un lenguaje **orientado a objetos**, lo que implica que se basa en la creación y manipulación de objetos. A diferencia de JavaScript, Java posee todas las características, herramientas y mecanismos para ser completamente orientado a objetos, como la encapsulación de datos y comportamientos en clases, herencia, polimorfismo y abstracción.
- Java es compatible con múltiples sistemas operativos y arquitecturas de hardware, es decir, es **multiplataforma**, lo que le permite ejecutarse en una amplia variedad de dispositivos, desde computadoras de escritorio y servidores hasta dispositivos móviles.
- Utiliza un sistema de recolección de basura (garbage collection) para **gestionar automáticamente la memoria** utilizada por los objetos. Esto simplifica el proceso de gestión de memoria y ayuda a prevenir errores comunes como fugas de memoria y el acceso a objetos eliminados.

¿Qué ventajas y desventajas tiene este framework?

Como se pudo analizar anteriormente, *Spring Boot* posee muchas características positivas, como la configuración automática del proyecto, evitando así utilizar componentes innecesarios, pero se debe considerar que la curva de aprendizaje del lenguaje de programación en el que se basa es alta y es difícil de dominar, dado que lleva cerca de 30 años en el mercado

Tabla 2: Ventajas y desventajas de Spring Boot
Fuente: Elaboración Propia.

Ventajas	Desventajas
Proporciona una configuración automática y una estructura de proyecto predefinida, reduciendo la cantidad de código y configuración a desarrollar.	Requiere un conocimiento sólido de Java y el ecosistema Spring. La curva de aprendizaje inicial puede ser pronunciada para aquellos nuevos en esta tecnología.
Utiliza Maven o Gradle para la gestión de dependencias, facilitando la incorporación y administración de bibliotecas y componentes externos necesarios.	Incluye dependencias preconfiguradas, lo que resulta en un aumento en el tamaño de la aplicación y la sobrecarga de recursos, afectando el rendimiento y la eficiencia.
Detecta dependencias y las configura automáticamente con las bibliotecas y componentes utilizados, reduciendo la configuración manual y mejorando la productividad.	Puede requerir configuraciones adicionales y complejas. Esto puede llevar tiempo y requerir un mayor nivel de experiencia.
Cuenta con una amplia gama de recursos, documentación, tutoriales y ayuda disponibles en línea. Además, tiene un historial establecido y una base de usuarios sólida, lo que garantiza soporte y actualizaciones.	Cuando el proyecto crece en tamaño, resulta desafiante mantener la estructura organizada y su gestión de configuraciones y dependencias, requiriendo cuidados en la arquitectura y planificación del proyecto.

y cuenta con una amplia comunidad, así como múltiples herramientas que puede ofrecer [Mira, 2020].

Spring Boot también consume mucho tiempo a la hora de hacer modificaciones en los proyectos, ya que se necesita un gran conocimiento en este framework para realizar un cambio, lo que lo vuelve engorroso para el programador [Roomi, 2022].

Además, al utilizar Spring Boot, estaremos utilizando una sintaxis en el lenguaje que no es cómoda para el programador, haciéndola menos legible en comparación con otros lenguajes. Como se muestra en la figura a continuación, donde comparamos el procedimiento de abrir un archivo, ejemplificando en el contexto en el que estamos, que tendremos que interactuar con diferentes archivos para la conversión de formatos, utilizando la menor cantidad posible de líneas de código.

En la imagen anterior, para el procedimiento de abrir un archivo en Java se requieren más líneas de código que en Python, teniendo una diferencia de 14 líneas para una simple acción. Esto, si lo llevamos a un nivel mucho más macro, hará que en Java se requiera una cantidad enorme de líneas de código al desarrollar el proyecto, lo que dificultará la legibilidad para el

Java

```java
File dir = new File("."); // get current directory
File fin = new File(
    dir.getCanonicalPath() + File.separator + "Code.txt"
);

FileInputStream fis = new FileInputStream(fin);

// Construct the BufferedReader object
BufferedReader in = new BufferedReader(new InputStreamReader(fis));

String aLine = null;
while ((aLine = in.readLine()) != null) {
    // //Process each line, here we count empty lines
    if (aLine.trim().length() == 0) {}
}

// do not forget to close the buffer reader
in.close();
```

Python

```python
my_file = open("/home/xiaoran/Desktop/test.txt")

print(my_file.read())
my_file.close()
```

Figura 16: Syntax Java vs Python
Fuente: Belitsoft 2023

programador y será menos cómodo y eficiente a la hora de programar.

En la imagen anterior se pudo observar que el framework mantiene la misma sintaxis que Java, ya que no ofrece ninguna simplificación de código dentro de las herramientas disponibles.

Finalmente, otro punto que nos lleva a descartar el uso de este framework en el proyecto es que el proceso de compilación y arranque de una aplicación Spring Boot es tremendamente lento, lo que aumenta considerablemente la integración. Por lo tanto, nuestra opción de utilizar Java Spring Boot queda descartada [Roomi, 2022].

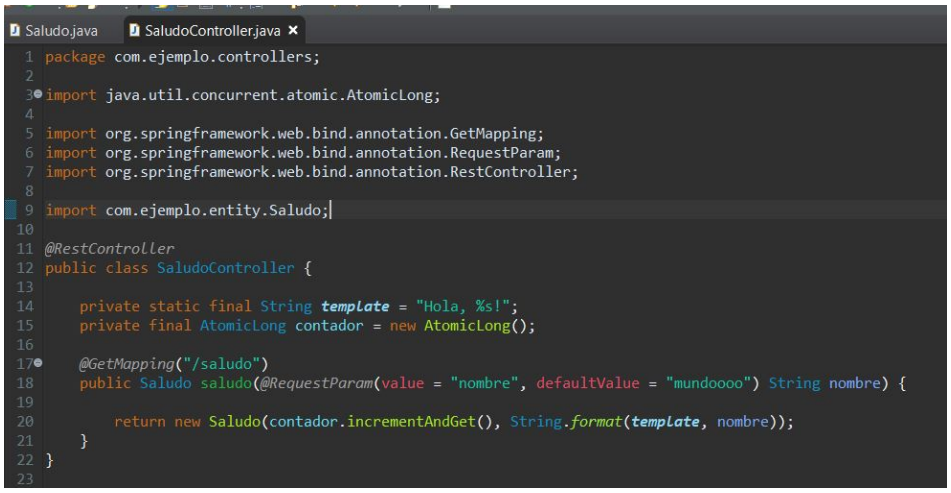


Figura 17: Ejemplo de una consulta GET en Spring Boot
Fuente: Sitio web de carloszr.com

3.5.3. Gin

En el contexto de Go (Golang), Gin es un framework web ligero y de alto rendimiento que se utiliza para construir aplicaciones web. Fue creado para ofrecer una solución simple y rápida para el desarrollo de servidores web en Go.

Gin sigue el enfoque de diseño minimalista y eficiente de Go, y se destaca por su velocidad y facilidad de uso. Aunque es un framework liviano, Gin proporciona una serie de características y funcionalidades esenciales para el desarrollo de aplicaciones web.

Una de las principales características de Gin es su enrutador, que permite definir rutas y manejar las solicitudes HTTP entrantes. Con Gin, puedes especificar rutas y asociarlas a funciones o controladores específicos que se ejecutarán cuando se acceda a esas rutas. Esto hace que el enrutamiento de las solicitudes sea muy sencillo y flexible.

Gin también incluye un sistema de middleware, que permite agregar capas adicionales de funcionalidad a tu aplicación. Los middleware en Gin son funciones que se ejecutan antes o después de que se maneje una solicitud HTTP. Pueden utilizarse para realizar tareas como la autenticación de usuarios, la autorización, el registro de solicitudes, la compresión de respuestas y más. Los middleware son una parte fundamental de Gin y te permiten personalizar el flujo de manejo de las solicitudes según tus necesidades [Golang, 2022].

Otra característica importante de Gin es su capacidad para manejar la validación de los datos de entrada. Proporciona funciones y estructuras que facilitan la validación de los parámetros de la solicitud HTTP, lo que te ayuda a garantizar que los datos recibidos sean correctos y seguros.

Además, Gin ofrece soporte para el renderizado de plantillas, lo que te permite generar con-

tenido HTML dinámico utilizando diversos motores de plantillas como HTML/template, Pongo2, Amber, entre otros.

```go
package main

import (
    "github.com/gin-gonic/gin"
)

func main() {
    r := gin.Default()
    r.GET("/ping", func(c *gin.Context){
        c.JSON(200, gin.H{
            "message": "pong",
        })
    })
    r.Run()
}
```

Figura 18: Contruyendo un endpoint que devuelve un mensaje en Gin
Fuente: Elaboración propia

Gin está basado en el lenguaje de programación Go, que principalmente es un lenguaje de programación compilado creado por la empresa Google en el año 2007 para la programación entre sistemas. Es similar al lenguaje de programación C, pero con una sintaxis más amigable para el desarrollador [Villa, 2022], contando con varias características:

- Go es muy eficiente en la compilación para generar código de máquina nativo, es decir, instrucciones ejecutables directamente por el procesador. Como resultado, es ideal para crear aplicaciones de alto rendimiento como servidores web o servicios en red que pueden gestionar un gran volumen de solicitudes sin ralentizarse.
- La concurrencia está integrada en Go a través de Goroutines y canales, lo que permite a los desarrolladores escribir programas que pueden ejecutar múltiples tareas o procesos simultáneamente. Esto hace que la creación de sistemas eficientes y escalables sea simple sin tener que lidiar con las complejidades de la gestión manual de subprocesos o procesos.
- Go tiene una sintaxis simple y concisa que es fácil de aprender y leer. La sintaxis es similar a C pero tiene ciertas características como la recolección de elementos no utilizados. Esto lo convierte en una excelente opción para los desarrolladores que son nuevos en la programación o que desean escribir un código limpio y fácil de mantener.
- Su sistema de tipos fuerte y estático ayuda en la prevención de errores de programación comunes como desreferencias de puntero nulo y desbordamientos de búfer. Esto lo convierte en una excelente opción para desarrollar software fiable y seguro, especialmente en sectores como el financiero y el sanitario, donde los errores de software pueden tener graves consecuencias.

- Go tiene una comunidad de desarrolladores grande y activa que contribuye al lenguaje y proporciona una biblioteca estándar de Go cada vez mayor de paquetes y herramientas de código abierto para ayudar a los desarrolladores a crear una amplia gama de aplicaciones. Se incluye todo, desde marcos web y controladores de bases de datos hasta bibliotecas de aprendizaje automático y herramientas de cadena de bloques.

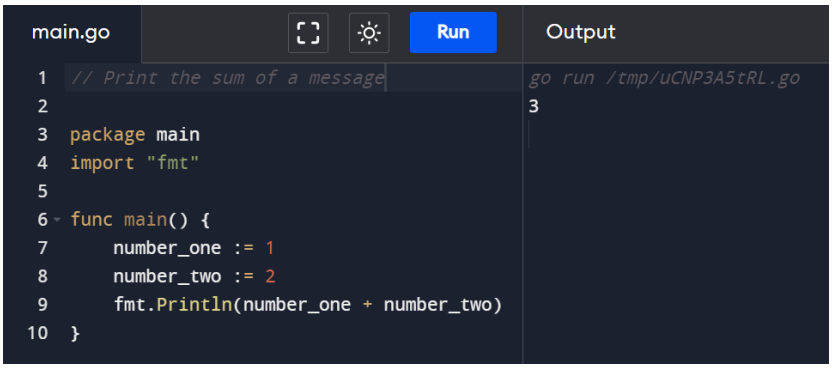


Figura 19: Realizando una suma en golang
Fuente: Hackernoon

Volvemos al framework Gin, mostrando las ventajas y desventajas de esta tecnología para saber si es el mejor candidato para desarrollar nuestra plataforma.

Tabla 3: Ventajas y desventajas de Gin
Fuente: Elaboración Propia.

Ventajas	Desventajas
Conocido por su alta velocidad y rendimiento. Está diseñado para ser rápido y eficiente, lo que lo convierte en una excelente opción para aplicaciones web de alto rendimiento.	Un marco relativamente fácil de aprender, puede tener una curva de aprendizaje si no se está familiarizado con el lenguaje Go, lo que puede llevar un tiempo adaptarse a su sintaxis y características.
Posee un marco minimalista y liviano. No tiene muchas dependencias y se enfoca en proporcionar solo las funcionalidades esenciales para el desarrollo web. Esto significa que es fácil de aprender, usar y mantener.	Su ecosistema de bibliotecas y complementos sigue siendo más pequeño en comparación con lenguajes más establecidos, como Python o JavaScript. Significa que hay menos bibliotecas o soluciones específicas para las necesidades en comparación con otros marcos de desarrollo web.
Ofrece un enrutamiento rápido y eficiente. Utiliza un enrutador de alto rendimiento y permite definir rutas de manera sencilla y clara, lo que facilita el manejo de las solicitudes HTTP y el direccionamiento de las rutas.	Gin Gonic tiene una documentación oficial, es posible que encuentres menos recursos y ejemplos en comparación con otros marcos de desarrollo web más establecidos. Esto puede dificultar un poco la curva de aprendizaje o la resolución de problemas más avanzados.

Una ventaja que nos favorece dentro de las características de este framework es el diseño rápido y eficiente que posee, lo que hace que la aplicación web desarrollada tenga un alto rendimiento, sobre todo a la hora de leer archivos que pueden tener un peso grande si el *XML* contiene muchas etiquetas, facilitando así la traducción de los diferentes tags de manera rápida.

Sin embargo, su ecosistema presenta problemas dado que no está tan desarrollado como los demás lenguajes, lo que hace que tengamos que construir las soluciones para nuestras necesidades. En otros lenguajes, estas soluciones ya están desarrolladas y listas para ser usadas, por lo que en Gin tendríamos que dedicar más tiempo del que podríamos destinarle a la construcción de software. Esto se suma a que su documentación oficial cuenta con menos recursos para solucionar problemas avanzados que podrían aparecer.

Gin, al estar hecho en base a Golang, utiliza la programación estructurada. Su estado se maneja a través de variables globales o pasándolas como argumentos a las funciones, a diferencia de la programación orientada a objetos. Además, también puede ocurrir la duplicación de código a la hora de querer escalar o hacer crecer el software, por lo que este framework

queda lejos de ser nuestra elección.

3.5.4. Django

Django es un framework de desarrollo web de código abierto y escrito en Python. Es un marco web Modelo-Vista-Template (MVT) que se utiliza para crear aplicaciones web. Se define a sí mismo como un marco web de ”baterías incluidas”, lo que implica que el framework proporciona una gran cantidad de componentes y características listos para usar, con robustez y simplicidad para ayudar a los desarrolladores web a escribir código limpio, eficiente y poderoso. Se encuentra entre los marcos web más famosos del mundo y también es uno de los marcos más utilizados. Es utilizado por Instagram, Youtube, Google e incluso la NASA para su sitio web. Así que vamos a desglosarlo aún más para aprender más al respecto.

Django sigue una arquitectura MVT que significa Model-View-Template. MVT es una variación de Django de la famosa estructura Modelo-Vista-Controlador, por eso sentirás que es bastante similar a cómo funcionan otros marcos. Cuando el servidor Django recibe una solicitud, el enrutador de URL asigna la solicitud a la vista adecuada. Luego, la vista obtiene los datos a través de los modelos, completa la plantilla y la envía de regreso al usuario.

La arquitectura que utiliza Django proporciona una capa de Object-Relational-Mapping (ORM) que simplifica el manejo y comunicación con la base de datos, acelerando el proceso de desarrollo. Sin él, los desarrolladores tendrían que escribir las consultas a mano, lo que generaría una gran cantidad de SQL complejo y difícil de mantener, eventualmente con problemas de seguridad [Romanos, 2023], por lo que en el contexto de este problema, no es prioritario invertir mucho tiempo haciendo definiciones .ª mano”.

La capa de Template o plantilla se utiliza para separar los datos de la forma en que realmente los presenta y ve el usuario. Está inspirado en DRY, que es un principio básico y ampliamente utilizado en el diseño de plantillas. Sus siglas significan ”Don’t Repeat Yourselfz su objetivo es permitir la reutilización mediante el uso de plantillas como la barra de navegación lateral, barra de navegación principal, encabezado de la página, etc. Esto minimiza la repetición y hace que la escritura de código sea más eficiente y limpia.

La Vista en Django es responsable de procesar la solicitud del usuario y enviar una respuesta válida. Obtiene los datos del modelo, da a cada plantilla acceso a datos específicos para mostrar, o podría realizar algún procesamiento sobre los datos de antemano. Hoy en día, las vistas de Django pueden ser funciones que procesan la solicitud y devuelven una respuesta, o pueden ser clases capaces de hacer mucho más de manera similar a los controladores de Laravel y Rails [Uniwebsidad, 2023].”

Este framework se basa en el lenguaje de programación Python, que fue creado por Guido van Rossum y lanzado por primera vez en 1991 [Wikipedia, 2023]. Python se caracteriza por su sintaxis clara y legible, lo cual facilita la lectura y comprensión del código, lo que a su

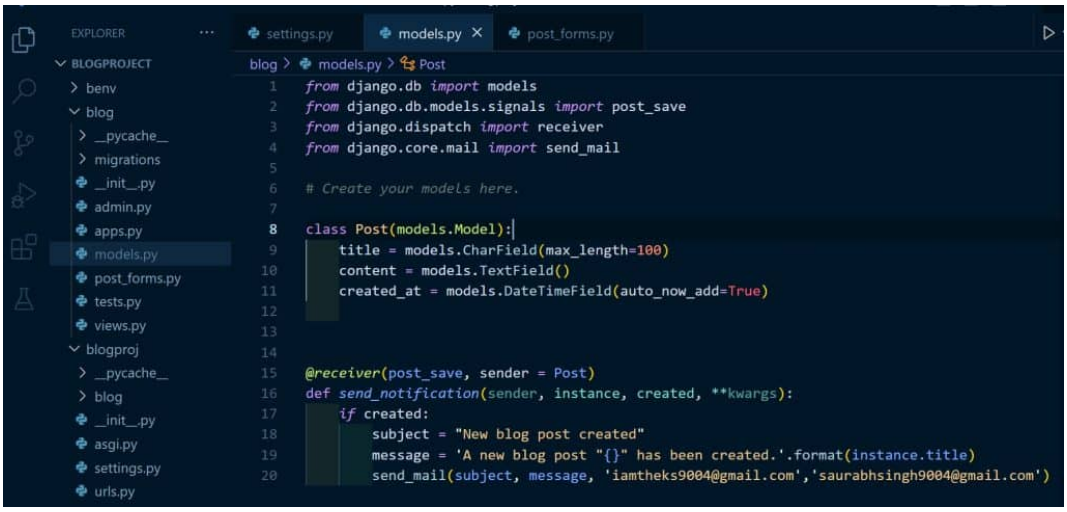


Figura 20: Ejemplo codigo en Django
Fuente: PythonGuía

vez favorece el desarrollo rápido y eficiente de aplicaciones además de poseer las siguientes características:

- Lenguaje de programación **multiplataforma**, que se refiere a aquel que puede ser utilizado en diferentes sistemas operativos o plataformas sin necesidad de realizar cambios significativos en el código. Esto significa que un programa escrito en un lenguaje multiplataforma puede ejecutarse en diversas plataformas, como Windows, macOS, Linux, entre otros, sin requerir modificaciones extensas.
- Lenguaje de programación **multiparadigma**, lo que significa que admite diferentes enfoques o paradigmas de programación. Un paradigma de programación define un conjunto de principios y técnicas para resolver problemas y estructurar el código.
- Se define por ser un **lenguaje de alto nivel**. El término .ªlto nivel"se refiere al nivel de abstracción y cercanía al lenguaje humano que tiene un lenguaje de programación. En el contexto de la programación, existen diferentes niveles de lenguajes, que van desde los lenguajes de bajo nivel, que son parecidos al lenguaje de máquina actuando directamente con el *hardware* [universidadviu, 2018], hasta los de alto nivel.
- Es **interpretado**, lo que significa que los programas escritos en Python no se traducen directamente a instrucciones de bajo nivel antes de su ejecución. En su lugar, son ejecutados por un intérprete de Python que lee y ejecuta el código fuente línea por línea en tiempo de ejecución.
- Es de **tipado dinámico**, lo que se refiere a una característica de ciertos lenguajes de programación en la cual las variables no están asociadas a un tipo de dato específico durante la declaración o asignación, sino que su tipo puede cambiar en tiempo de

ejecución. Las variables pueden almacenar valores de diferentes tipos en diferentes momentos del programa. No es necesario declarar explícitamente el tipo de una variable antes de usarla, ya que el tipo se determina en tiempo de ejecución según el valor asignado.

Ahora la pregunta que debemos hacernos es ¿Por qué es una parte importante el tipado dinámico para la solución de nuestro problema? Como se puede observar en la imagen presentada a continuación, el tipado estático, que es lo opuesto al tipado dinámico, se refiere a que una vez que hayas definido una variable, no podrás cambiar la procedencia de donde esta venga. Quiere decir que si la defines como String, no puedes asignarle un valor numérico o *booleano (True or False)*, ya que la compilación de este no podrá ejecutarse correctamente.

Figura 21: Ejemplo de tipificado estático
Fuente: Elaboración Propia

Ahora, en esta segunda imagen presentada, vemos que declaramos una variable con contenido de tipo String, pero luego podemos definirla nuevamente con otro contenido de tipo numérico.

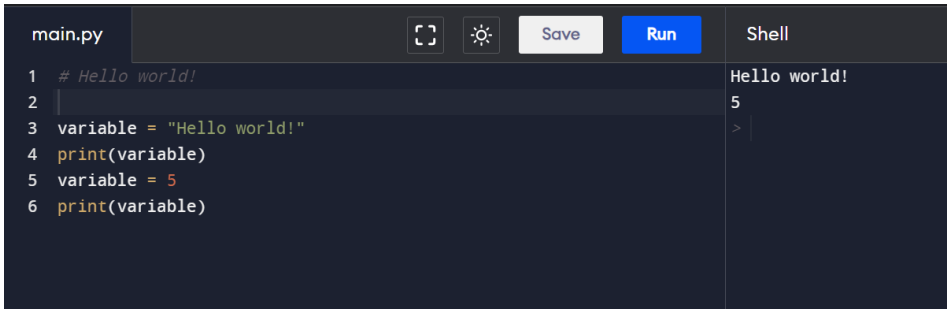


Figura 22: Ejemplo de tipificado dinámico
Fuente: Elaboración Propia

La información anteriormente dada nos indica que Python es el lenguaje perfecto para poder desarrollar nuestra aplicación web, pero no debemos olvidar que aún falta por ver el framework en el que está basado este lenguaje, donde debemos observar las ventajas y desventajas con las que nos encontraremos al estar construyendo el software.

Tabla 4: Ventajas y desventajas de Django
Fuente: Python Geeks

Ventajas	Desventajas
Suficientemente poderoso y versátil que ayudará a los desarrolladores a construir API web escalables.	El marco tiene muchas configuraciones y opciones que los desarrolladores deben ajustar. Aunque es un marco simple y fácil de entender, a menudo se afirma que es difícil de dominar.
Django sigue la filosofía de baterías incluidas, lo que significa que el marco tiene todo lo que necesita para desarrollar su aplicación web. Con una gran comunidad de código abierto, también es más fácil encontrar paquetes que se ajusten a sus necesidades.	Los componentes de Django están estrechamente acoplados, lo que obliga a los desarrolladores a instalar todos los componentes al mismo tiempo. Esto puede ser molesto y lento al migrar sus proyectos en los sistemas.
La arquitectura del marco aborda problemas de seguridad comunes como el secuestro de clics, las vulnerabilidades de SQL y los scripts entre sitios.	Django tiene una gran cantidad de código que requiere potencia informática del servidor y puede tardar mucho tiempo, lo que afecta a las aplicaciones pequeñas.
Django proporciona una interfaz de administrador integrada que facilita mucho a los desarrolladores el manejo de la base de usuarios al proporcionar controles de acceso sin tener que escribir el código de administrador desde cero.	Siendo un marco web moderno, no admite el manejo de múltiples solicitudes simultáneamente. Por lo tanto, los desarrolladores tienen que idear diferentes enfoques para hacer frente a múltiples solicitudes rápidamente.

Con todo lo anterior, podemos notar que Django ofrece interesantes herramientas para construir aplicaciones web. Sin embargo, carece de herramientas para manejar archivos, tareas en segundo plano y operaciones en tiempo real, lo que lo hace menos atractivo para nuestro proyecto. Además, su curva de aprendizaje es pronunciada, lo que puede dificultar la adopción de este framework para desarrollar nuestra solución. Si bien se pueden realizar mediante módulos de Python, no vienen integrados en Django, lo que hace que el desarrollo sea más lento y menos eficiente.

3.5.5. Ruby On rails

Tenemos el último framework para analizar y ver si es la mejor herramienta para desarrollar la solución final: Ruby on Rails. Esta herramienta, también conocida como Rails, es un framework de desarrollo web de código abierto escrito en el lenguaje de programación Ruby. Fue creado por David Heinemeier Hansson y lanzado por primera vez en 2004. Rails sigue el patrón de diseño Modelo-Vista-Controlador (MVC) y proporciona una estructura y convenciones predefinidas para desarrollar aplicaciones web [Baumann, 2021].

Rails se basa en varios principios clave, como la convención sobre la configuración, lo que significa que el desarrollador solo necesita especificar las opciones no convencionales, ya que Rails asume las convenciones y proporciona configuraciones predeterminadas, lo que facilita el desarrollo al eliminar la necesidad de configurar explícitamente muchos aspectos de la aplicación [Hansson, 2023]. Esto permite un desarrollo rápido y eficiente al minimizar la cantidad de código repetitivo y redundante.

Una de las características importantes es la capa de abstracción de base de datos de Rails (ActiveRecord), que facilita la interacción con la base de datos y permite realizar operaciones de creación, lectura, actualización y eliminación (CRUD) de manera sencilla y elegante.

Así como también existe Active Record, Rails ofece muchas otras interfaces para operaciones comunes, tales como Active Storage para el manejo de archivos, Action Cable para el manejo de operaciones en tiempo real, Action Mailer para el envío de correos, Active Job para el manejo de tareas en segundo plano, entre otros. Esto permite a los desarrolladores aprovechar una amplia gama de funcionalidades y herramientas integradas en Rails para abordar diferentes requisitos y escenarios de desarrollo de una forma extremadamente ágil, destacándose ampliamente frente a sus competidores.

Otra de las características importantes presentes en la comunidad de Rails son sus gemas. Ruby utiliza un sistema de gestión de paquetes de Ruby (RubyGems), y Rails hace un amplio uso de las gemas para extender sus funcionalidades. Hay una amplia variedad de gemas disponibles para abordar diferentes requisitos y funcionalidades, lo que permite a los desarrolladores agregar características adicionales a sus aplicaciones de manera sencilla.

Ruby on Rails ha ganado popularidad debido a su enfoque en la simplicidad y la productividad, y ha sido utilizado para desarrollar una amplia gama de aplicaciones web exitosas. Es especialmente adecuado para el desarrollo de aplicaciones web basadas en bases de datos, y su comunidad activa y próspera ofrece una gran cantidad de recursos y soporte para los desarrolladores.

Rails está hecho en Ruby, que es un lenguaje de programación interpretado y de alto nivel creado por Yukihiro Matsumoto, también conocido como Matz, en Japón a mediados de la década de 1990. Ruby se destaca por su simplicidad, elegancia y enfoque en la productividad del desarrollador. A continuación se presentan algunas de las características más importantes de Ruby:

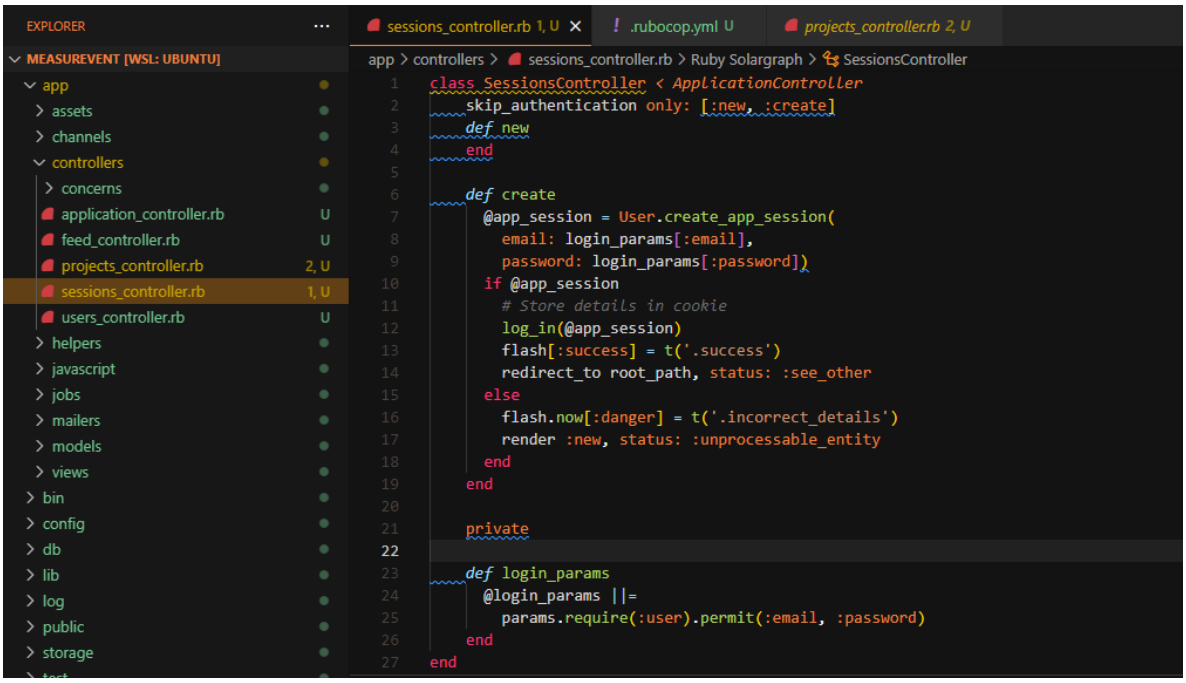
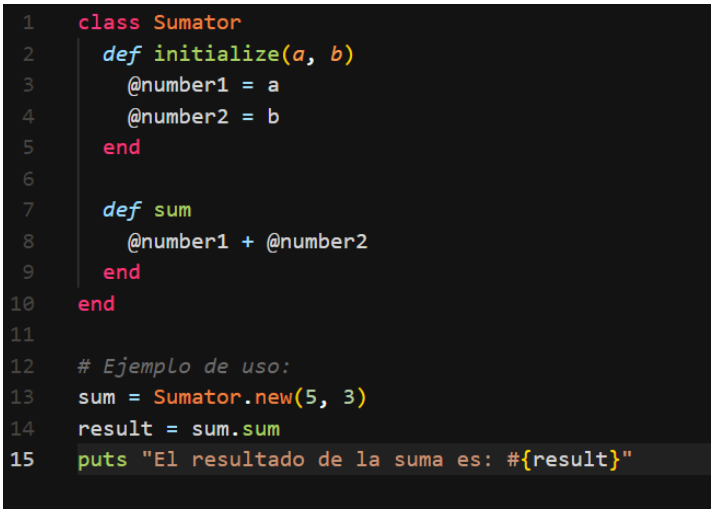


Figura 23: Ejemplo codigo en Rails
Fuente: Elaboración propia

- Ruby tiene una sintaxis clara y legible, que se asemeja al **lenguaje natural**, y se enfoca en la facilidad de lectura y comprensión del código. Esto facilita el aprendizaje y la escritura de programas en Ruby.
- Es **interpretado**, es decir, como lo hemos dicho anteriormente, que se necesita que el intérprete de Ruby analice el código y lo traduzca en lenguaje de máquina entendible por un ordenador, pero no existe un proceso previo de compilación como en C o Java [mytaskpanel, 2023].
- Es un lenguaje completamente **orientado a objetos**, lo que significa que todo en Ruby es un objeto. Cada valor y cada pieza de código se trata como un objeto, lo que permite la reutilización de código y la aplicación de conceptos de programación orientada a objetos de manera eficiente.
- Ruby es un lenguaje de tipado dinámico [Wikidot, 2022]. Esto brinda flexibilidad y facilita la escritura de código genérico y reutilizable.
- Cuenta con una **biblioteca o librería estándar** rica y extensa que ofrece una amplia gama de funcionalidades y herramientas para diversas tareas de programación, como manejo de archivos, manipulación de cadenas, acceso a bases de datos, networking y mucho más [Paramio, 2011]. Esto permite a los desarrolladores ahorrar tiempo al tener muchas funciones y utilidades disponibles de forma predeterminada.

- Ruby ofrece una característica llamada bloques, que se definen como fragmentos de código que se pueden pasar como argumentos a métodos. Los bloques permiten escribir código más conciso y expresivo, y se utilizan ampliamente en Ruby para iterar sobre colecciones de datos. Pueden capturar el estado y formar closures, lo que los hace aún más poderosos [Daniel-Penaloza, 2021].
- Es **open source y multiplataforma**, ya que se puede descargar gratis de la página oficial y ejecutarlo en diferentes sistemas operativos [HostingPlus, 2021].



```ruby
class Sumator
  def initialize(a, b)
    @number1 = a
    @number2 = b
  end

  def sum
    @number1 + @number2
  end
end

# Ejemplo de uso:
sum = Sumator.new(5, 3)
result = sum.sum
puts "El resultado de la suma es: #{result}"
```

Figura 24: Sumando 2 numeros con clases en Ruby
Fuente: Elaboración propia

¿Qué ventajas y desventajas presenta este framework?

Tabla 5: Ventajas y desventajas de Rails
Fuente: Elaboración propia

Ventajas	Desventajas
Sigue el principio de çonvención sobre configuración", lo que significa que proporciona una estructura y un conjunto de convenciones que facilitan el desarrollo rápido de aplicaciones web permitiendo escribir menos código y ser más productivos.	Aunque puede ser considerado fácil de aprender, su curva de aprendizaje puede ser empinada para desarrolladores nuevos en el mundo de los frameworks, debido a su amplia gama de conceptos y convenciones.
Tiene una sintaxis elegante y legible, lo que facilita la comprensión y mantenimiento del código, aumentando el desarrollo del proyecto.	Al ser un lenguaje interpretado puede afectar el rendimiento en comparación con lenguajes más rápidos como **C++** o **Java**.
Cuenta con una comunidad activa y apasionada que ha desarrollado una amplia gama de gemas y herramientas, aprovechando soluciones existentes y acelerando el desarrollo de nuevas características [Ruby, 2023].	Si bien se ha desarrollado una gran cantidad de gemas útiles, puede haber ciertas dependencias a la hora de actualizar una gema y producir cierta incompatibilidad en el proyecto.

Las ventajas de convención sobre configuración hacen que se destaque Rails por sobre los otros frameworks, ya que los demás frameworks aún no están lo suficientemente maduros para poder tener listas las funcionalidades que se requieren en cada proyecto y sobretodo las buenas prácticas necesarias, como el uso de ORM, la separación del modelo-vista-controlador, así como también la facilidad de tener un sistema de tareas en segundo plano integradas, junto con un manejo de archivos y operaciones en tiempo real.

3.6. Elección del framework

Ya en las subsecciones anteriores pudimos describir y explicar cada framework popular y de carácter maduro para el desarrollo de nuestra solución junto con sus respectivas ventajas y desventajas para saber si puede ser una buena opción a la hora de desarrollar una escalabilidad y mantenimiento del software a futuro.

Sin embargo, el framework elegido ha sido **Ruby On Rails**. Todo esto en base a la investigación realizada anteriormente, comparando cada punto de las herramientas elegidas. Este framework tiene la característica principal de **productividad y convención sobre configura-**

ción, lo que hace que un desarrollador, en vez de estar ocupando su tiempo configurando la estructura del proyecto, se centre solamente en la lógica de negocios. La configuración del proyecto viene por defecto, facilitando el desarrollo ágil, mientras que en los otros frameworks se debe comenzar todo desde cero.

Agregando al punto anterior, también se debe considerar la **amplia comunidad y madurez** que, en el caso de Ruby On Rails, se ha estado desarrollando durante más de dos décadas, lo que termina resultando en una gran cantidad de bibliotecas, gemas y recursos disponibles para la resolución de problemas y búsqueda de ayuda. Incluso el propio creador de este framework ha creado una fundación con la finalidad de poder llegar a más personas.

Otro punto importante a considerar es la **escalabilidad y rendimiento adecuado**. Puede ser que no sea lo suficientemente conocido por el rendimiento extremo como algunos otros frameworks, pero es lo suficientemente escalable para manejar aplicaciones medianas y grandes. Además, cada año Rails ha estado mejorando su rendimiento y la velocidad de su herramienta, haciendo que en cada versión que salga a la luz sea mucho mejor.

De la misma manera, el hecho de que Rails tenga tantas herramientas listas para usar, como la gestión de archivos, tareas en segundo plano, operaciones en tiempo real, entre otras, hace que sea una herramienta muy completa y que se adapte a las necesidades de nuestro proyecto con excelencia.

Cada una de las razones mencionadas anteriormente fue basada en la evaluación exhaustiva de los requisitos específicos del proyecto y las metas a largo plazo. Node.js, Spring Boot, Django y Golang también son excelentes frameworks con sus propias fortalezas en términos de escalabilidad, rendimiento y comunidad de desarrollo. Por lo tanto, siempre es recomendable realizar una investigación profunda, como la que fue realizada dentro de esta sección completa.

3.7. Historial de funcionamiento

3.7.1. ¿Por qué almacenar historial de uso?

Cuando hablamos del historial de funcionamiento, uno se tiene la idea de que se pueda referir a tener algún tipo de registro de todos los hechos o funcionalidades que hará la plataforma cuando se necesite transformar o convertir desde un tipo de archivo a otro. Pues esta idea está totalmente en lo correcto; se necesita tener un historial acerca de todos los procedimientos o hechos que se han realizado, todo esto con el sentido de que, si en algún momento un usuario tiene un mensaje de error o la aplicación no funciona por algún motivo, entonces esto nos ayudará a arreglar errores que no se hayan descubierto en la fase de validación de la solución, la cual se mostrará detalladamente en el siguiente capítulo. Esto, naturalmente, nos permitirá mejorar los servicios que ofrece la plataforma. Además, esto

también puede ser utilizado para realizar ciencia de datos, que es un estudio con el fin de extraer información significativa para una institución o empresa [Amazon, 2023].

Estos registros son llamados registros *log*, que proporcionan información muy importante sobre las actividades implícitas y explícitas de cualquier sistema de hardware y software informático. Son un tipo de archivos que contienen mensajes sobre el sistema, incluyendo el kernel, los servicios y las aplicaciones en ejecución.

3.7.2. Almacenamiento de la información

La pregunta que se debe hacer en estos momentos es dónde se deberían almacenar todos estos registros. Si bien se podrían almacenar en un archivo de texto plano, también tenemos la posibilidad de almacenarlos en una base de datos. Pero ¿por qué elegir uno y no el otro? Debemos analizar bien qué características tiene cada uno y qué los hace especiales en comparación con el otro.

1. Rendimiento y escalabilidad
    - **Archivo de texto:** Es mucho más rápido y eficiente de escribir debido a la simplicidad que tiene realizar esta acción. Sin embargo, cuando el volumen aumenta, puede volverse mucho más lento analizar grandes cantidades de archivos.
    - **Base de datos:** Una base de datos está hecha para almacenar grandes cantidades de datos y, si está bien optimizada, puede manejar grandes cantidades de logs y permitir búsquedas y consultas eficientes. Es mucho más escalable a medida que crece la cantidad de datos.
2. Búsqueda y análisis:
    - **Archivo de texto:** La búsqueda de información en un archivo de texto puede ser más compleja y demorosa, ya que no hay una estructuración fija de los datos en el archivo, especialmente cuando hay grandes cantidades de datos en un mismo archivo.
    - **Base de datos:** Permite realizar consultas más complejas y filtrarlas con mayor facilidad debido a que cuentan con una estructuración y herramientas necesarias para realizarlas.
3. Respaldo y recuperación:
    - **Archivo de texto:** Los archivos de texto pueden ser más sencillos de copiar o mover, pero en caso de pérdida o daño, la información se perderá permanentemente.
    - **Base de datos:** Las bases de datos cuentan con sistemas de respaldo y recuperación en caso de fallas o pérdidas de la información en los sistemas.

4. Tamaño del almacenamiento:
    - **Archivo de texto:** Los logs en archivos de texto tienden a ocupar menos espacio de almacenamiento.
    - **Base de datos:** Puede requerir mucho más espacio debido a la estructuración de la información en las bases de datos.

En conclusión, sabemos que un archivo de texto es una buena opción para almacenar información debido a su bajo espacio de almacenamiento y facilidad para mover archivos de una parte a otra. Sin embargo, hay que considerar que la información, al no estar estructurada, puede ser más lenta en el proceso de lectura, mientras que una base de datos estructurada facilita la manipulación de los datos y ofrece opciones de filtrado que optimizan la búsqueda de información, además de contar con la capacidad de recuperar la información en caso de alguna falla.

3.7.3. Bases de datos

Como ya se ha decidido, surge otra incógnita: ¿cuál base de datos utilizar para almacenar la información mencionada anteriormente? Para ello, es necesario analizar cada una de las bases de datos existentes y determinar cuál se ajusta más al contexto del problema que se quiere solucionar. Si bien todas las bases de datos tienen el objetivo de almacenar grandes cantidades de datos, se debe tener en cuenta que cada una cuenta con características propias que podrían facilitar la vida no solo durante el desarrollo del software, sino también una vez terminado.

Para este análisis, se extraerá la información de un sitio web llamado db-engines, que se encarga de rastrear y recopilar datos relevantes sobre una amplia variedad de bases de datos, tanto relacionales como no relacionales, presentándolas en forma de clasificaciones y gráficos. El objetivo principal de esta plataforma es ayudar a los desarrolladores, ingenieros de datos y profesionales de IT a tomar decisiones más informadas a la hora de seleccionar su base de datos.

En el 2023, el sitio web proporcionó un listado con todas las bases de datos más influyentes en el mundo, donde se puede ver en primer lugar a **Oracle**, en segundo lugar a **MySQL**, en tercer lugar a **Microsoft SQL Server** y en cuarto lugar a **PostgreSQL**.

Rank			DBMS
Jul 2023	Jun 2023	Jul 2022	
1.	1.	1.	Oracle
2.	2.	2.	MySQL
3.	3.	3.	Microsoft SQL Server
4.	4.	4.	PostgreSQL
5.	5.	5.	MongoDB
6.	6.	6.	Redis
7.	7.	7.	IBM Db2
8.	8.	8.	Elasticsearch
9.	9.	9.	Microsoft Access
10.	10.	10.	SQLite

Figura 25: Bases de datos mas populares en el mundo
Fuente: db-engine.

Aunque existan más bases de datos para poder analizar y buscar la mejor opción, acotaremos los candidatos para hacer esta búsqueda más eficiente y no invertir demasiado tiempo en ella. Si bien es importante la base de datos, ya que almacena la información que nos será útil en el futuro para manejar estadísticas, el problema está más enfocado en la parte algorítmica de la conversión de los archivos.

A continuación, se explicará cada una de las bases de datos especificando sus ventajas y desventajas más importantes:

- **Oracle:** Es un sistema de gestión de bases de datos relacionales muy utilizado en empresas y organizaciones de todo el mundo.
  - **Ventaja:** Su escalabilidad y rendimiento lo hacen esencial para manejar grandes volúmenes de datos y cargas de trabajo complejas. Gracias a su arquitectura, permite que esta DB se ajuste a las demandas de aplicaciones de alto rendimiento y crecimiento empresarial [Oracle, 2023].
  - **Desventaja:** El principal inconveniente es su alto costo debido a la adquisición inicial de la licencia y el mantenimiento continuo, lo que actúa como una gran barrera para las empresas más pequeñas.
- **MySQL:** Es un gestor de bases de datos relacionales de código abierto conocido por ser rápido, confiable y fácil de usar, por lo que se ha convertido en una opción popular para diferentes proyectos.
  - **Ventaja:** Es una base de datos de código abierto, por lo que es gratuito para su uso y distribución, lo que lo hace atractivo para diferentes proyectos empresariales.

  - **Desventaja:** Si bien MySQL es un motor de base de datos probado altamente en batalla en proyectos tan grandes como Facebook, algunas tareas de administración podrían ser complicadas, tales como las copias de seguridad automatizadas. La herramienta oficial que proporciona esto no es de código abierto y se debe acudir a soluciones de terceros como Percona para poder realizar esta tarea.

- **Microsoft SQL Server:** Es un sistema de gestión de bases de datos relacionales desarrollado por Microsoft. Presta servicio a aplicaciones de software destinadas a la inteligencia empresarial y análisis en entornos corporativos [Pérez, 2021].

  - **Ventaja:** Su ventaja más destacada es su estrecha integración con otros productos de Microsoft, lo que facilita la implementación y el mantenimiento en los entornos que utilizan tecnologías de Microsoft.
  - **Desventaja:** Su costo puede ser un obstáculo, especialmente para las grandes implementaciones empresariales, debido a las licencias y opciones de escalado costosas.

- **PostgreSQL:** Es un sistema de gestión de bases de datos relacionales de código abierto, muy conocido por su gran robustez, capacidad y compatibilidad con el estándar SQL, lo que lo hace popular para proyectos complejos y grandes volúmenes de datos.

  - **Ventaja:** Ofrece una amplia gama de características avanzadas que lo hacen adecuado para proyectos complejos, proporcionando gran flexibilidad y adaptabilidad para distintos escenarios. Su arquitectura extensible permite que los desarrolladores puedan crear sus propias extensiones y funciones que pueden personalizarse para necesidades específicas.
  - **Desventaja:** Debido a que tiene amplias funcionalidades en comparación con otras bases de datos, puede tener una curva de aprendizaje más pronunciada. Sin embargo, una vez que se tiene la experiencia adecuada, su uso puede ser altamente beneficioso.

Cada una de estas bases de datos tiene importantes ventajas que pueden resultar complementarias para el desarrollo de una aplicación web; sin embargo, también se deben tomar en cuenta las características negativas que opacan sus funcionalidades más significativas y que influyen en la decisión de implementar la base de datos.

Microsoft SQL Server y Oracle son de pago, y la idea no es incurrir en gastos cuando no se generan o no está contemplado obtener ganancias aún. Por lo tanto, estas dos opciones están descartadas. Aún nos quedan las alternativas de utilizar PostgreSQL y MySQL.

MySQL es rápido para realizar consultas de manera optimizada y es fácil de usar, lo que nos ahorra tiempo de trabajo. Sin embargo, algunas funciones son difíciles de administrar lo que podría llevarnos a tener problemas en el futuro de mantenibilidad y costos.

Por último, tenemos la base de datos PostgreSQL, donde su gran flexibilidad hace que cualquier proyecto que se quiera desarrollar sea 100 % compatible con esta herramienta. A pesar de tener una curva de aprendizaje más alta en comparación con las demás, eso no sería un gran impedimento para poder utilizarla, ya que, a medida que más la utilicemos y la configuremos según nuestras preferencias, más aprenderemos. Por lo tanto, PostgreSQL es la mejor opción para este proyecto.

3.8. Trabajos en segundo plano para el procesamiento de archivos.

Como ya hemos mencionado en todo este informe, usaremos diferentes tipos de archivos para procesar la información que necesitamos para alcanzar nuestro objetivo final. Sin embargo, debemos tomar en cuenta que estamos manejando archivos, lo que significa que el contenido y la estructura de estos, según lo que hemos definido anteriormente, puede ser muy complejo, ya que no se sabe a ciencia cierta qué es lo que el usuario nos va a entregar.

Es por esto que se ha decidido también implementar un sistema de colas, de manera que cada usuario que suba sus archivos para generar el proceso pueda ser ingresado a un sistema que administre o coordine estos ”mensajes”para ser posteriormente ingresados a distintos ”trabajadores”. Esto nos trae muchos beneficios, tales como:

- **Manejo de Alta Carga:** Un sistema de colas distribuye eficientemente la carga de trabajo entre múltiples trabajadores, que también puede ser ajustado dinámicamente, lo que permite procesar una gran cantidad de archivos simultáneamente sin sobrecargar el sistema.
- **Tolerancia a fallos:** Si un trabajo falla debido a un error temporal, el sistema de colas puede reintentarlo automáticamente, mejorando la resiliencia de la plataforma. Las fallas o errores en el procesamiento de un archivo no afectarán a otros archivos, ya que cada tarea se maneja de manera independiente.
- **Optimización de recursos:** Permite distribuir eficientemente los recursos del servidor, utilizando ciclos de CPU y memoria de manera más efectiva, asignando también prioridades a diferentes tipos de tareas, garantizando que las tareas más críticas se procesen primero.
- **Mejora la experiencia de usuario:** Al usar un sistema de colas, puedes proporcionar una respuesta inmediata al usuario indicando que su archivo ha sido recibido y está en proceso, mejorando la percepción de rapidez y eficiencia.

Anteriormente, decidimos utilizar **Ruby on Rails** para desarrollar nuestra plataforma. Este *framework* cuenta con un sistema de trabajos en segundo plano integrado llamado **Active Job** que utilizaremos para realizar los procesos de traducción.

3.9. Explicación del algoritmo

En la parte de investigación de otras soluciones similares a las que se desea desarrollar, se pudo concluir que la solución a desarrollar es única. Por lo tanto, el algoritmo que se debe construir debe ser creado al 100 % por el desarrollador. En esta sección, explicaremos a fondo y con detalles todo el proceso de construcción de la solución para una correcta conversión del formato de los archivos, con el fin de obtener una solución concreta.

3.9.1. Visión general

Para dar una mejor explicación del código, lo primero que tenemos que considerar es el camino que toma el usuario para realizar la conversión o traducción de los archivos correspondientes, es decir, cuáles son los pasos a seguir para que la plataforma pueda traducir los archivos y tener la conversión deseada. Para ello, son tres los archivos importantes:

- **Archivo de traducción:** Este tipo de archivo contiene todas las etiquetas XML que serán representadas finalmente como código LaTeX. Es importante destacar que solo contiene los *tags* que serán visualizados posteriormente en el resultado final, ya sea en LaTeX o PDF.
- **Archivo XML:** Este es uno de los archivos más importantes porque contiene todo el contenido de etiquetas XML que será posteriormente traducido gracias al archivo de traducción.
- **Archivo ZIP:** Este formato de compresión de archivo es necesario ya que contiene todos los ficheros que se requieren para compilar la traducción final, ya sean imágenes, archivos complementarios, etc., para así poder representar nuestro LaTeX final.

Uno de los archivos más importantes es el archivo XML, ya que es la base principal de todo este proceso de desarrollo y el que contiene toda la información que es entregada por el usuario.

Otro de los archivos importantes y que deberían estar presentes a la hora de realizar el proceso de conversión son el **Archivo de traducción** y el **Archivo XML**, ya que son parte fundamental para este proceso. Sin un archivo de traducción, no se podría saber a ciencia cierta qué son los *tags* que requiere el usuario para su informe final, además de cuáles son los comandos precisos de LaTeX que representarán a cada etiqueta. Esto es casi imposible de saber, ya que dependerá exclusivamente de la necesidad que tenga cada usuario en sí.

Por último, en cuanto al archivo ZIP de plantilla, este no será obligatorio, ya que la plataforma, en caso de no contar con una plantilla definida por el usuario con un preámbulo y postámbulo

que fuera creado o desarrollado por este mismo, realizará la conversión con la plantilla que viene por defecto en la plataforma, pudiendo así descargar el código LaTeX junto con su PDF.

Ahora que analizamos e hicimos una pequeña introducción sobre la base importante de esta sección, podemos entrar en profundidad analizando cada una de las funcionalidades creadas para llevar a cabo este proceso con éxito.

3.9.2. Algoritmo previo a traducción

Para el algoritmo de traducción lo primero que haremos es verificar si existe un archivo ZIP adjunto que tendrá la plantilla necesaria para almacenar la información traducida del XML.

```
class Translation
  ...
  def decompress_template_if_provided
    if translation.zip_file.attached?
      translation.zip_file.open do |zip_tempfile|
        @unzipped_dirname = File.join(File.dirname(zip_tempfile.path),
    SecureRandom.alphanumeric)
        system('unzip', zip_tempfile.path, '-d', @unzipped_dirname)
        tex_path = `find "#{@unzipped_dirname}" -name "main.tex"`
        if File.basename(tex_path.strip) != "main.tex"
          translation.update!(error_message: 'No se encontró el archivo
    main.tex en el zip', error_code: 'FILE NOT FOUND')
          raise
        end
        translator.template_dirname = File.dirname(File.join(tex_path.
    strip))
      end
    end
  end
  ...
end
```

Figura 26: Decompress ZIP File, Translation Class
Fuente: Elaboración propia.

Lo primero que haremos será verificar si existe tal archivo ZIP. En el caso de que nuestro archivo exista, lo descomprimiremos en el disco en la carpeta `/tmp/` y, dentro de esta carpeta, habrá otra carpeta con un nombre aleatorio de manera que no haya colisión de nombres si es que dos usuarios suben el mismo archivo con el mismo nombre. Para descomprimir correctamente el archivo, es necesario la existencia del archivo `main.tex` dentro del archivo zip, que, en caso de no venir adjunto, mostrará un mensaje de error.

El siguiente paso será invocar el método `translate` de la clase `Translator`. Una vez que haya traducido todos los archivos, o no, generaremos de todas formas una transacción para

mantener la atomicidad de los pasos posteriores a realizar. En caso de resultar con éxito, se adjuntará el PDF y el archivo ZIP generados por el método. En caso contrario, se mostrará un mensaje de error de acuerdo a la falla específica generada.

```
class Translation
...
  def translate
    if translator.translate
      translation.transaction do
        translation.translated_file.attach(io: File.open(translator.translated_file_path), filename: File.basename(translator.translated_file_path))
        translation.translated_zip_file.attach(io: File.open(translator.translated_zip_file_path), filename: translated_zip_filename)
        translation.completed!
      end
    else
      translation.transaction do
        translation.failed!
        translation.update!(error_message: translator.error_message, error_code: translator.error_code)
      end
    end
  end
...
```

Figura 27: Translate Files, Translation Class
Fuente: Elaboración propia.

Como último método de esta clase, tenemos la eliminación o destrucción del archivo descomprimido, que, al ser terminada la transacción de traducción, ya no nos servirá a futuro, por lo que es una buena práctica eliminar la ”basura”generada por nuestro código para así también evitar acumular espacio en el disco.

```
class Translation
...
  def remove_uncompressed_directory
    FileUtils.rm_rf(@unzipped_dirname) if @unzipped_dirname
  end
...
```

Figura 28: Remove Files Translation Class
Fuente: Elaboración propia.

3.9.3. Algoritmo de traducción

Si bien en la sección anterior vimos de manera más general cómo funcionaba la traducción de forma macro, lo que veremos en esta sección será la traducción misma del archivo en sí, es decir, tomando los archivos XML y el archivo de traducción entregados por el usuario que utiliza la plataforma. Para ello, se utilizó una clase `Translator` que, como su nombre lo indica, es la encargada de estructurar la información del XML a código LaTeX.

El primer paso a realizar es ver qué tipo de *Template* utilizar para el archivo LaTeX final. En el caso de que el archivo ZIP no sea proveído por el usuario, el sistema automáticamente dejará una plantilla por defecto.

```
class Translator
  ...
  def use_default_template_if_not_provided
    return if template_dirname.present?

    @template_dirname = File.join(Dir.tmpdir, SecureRandom.alphanumeric)
    Dir.mkdir(template_dirname)
    FileUtils.chdir template_dirname do
      latex_code = latex_code = "\\documentclass{article}
      \\usepackage[utf8]{inputenc}
      \\title{LaTeX Generator}
      \\author{Informe Análisis Forense}
      \\date{21/06/2023}
      \\begin{document}
      \\maketitle
      \\end{document}
      "
      xml_files.each do |xml_file|
        latex_code = latex_code.gsub("\\end{document}", "\\input{#{
   xml_file.filename}.tex}\n\\end{document}")
      end
      File.open('main.tex', 'w') { |file| file.write(latex_code) }
    end
  end
  ...
end
```

Figura 29: Which template to use?, Translator Class
Fuente: Elaboración propia.

Como se puede apreciar en el código, en caso de existir el archivo ZIP, no haremos ninguna acción, mientras que, si no existe tal archivo, generaremos una plantilla por *default* que contendrá todos los archivos XML subidos por el usuario en el documento.

Luego de haber verificado la existencia de una plantilla por defecto, procesaremos los archi-

vos XML y el archivo de traducción con el siguiente método:

```
class Translator
  ...
  def process_xml_files
    data = read_txt_file(txt_file)
    xml_files.each do |xml_file|
      xml_doc = Nokogiri::XML(xml_file.download)
      latex_code = clear_latex(translate_xml_to_latex(xml_doc, data))
      create_xml_translated_tex_file(xml_file, latex_code)
    end
  end
  ...
end
```

Figura 30: Process XML Files, Translator Class
Fuente: Elaboración propia.

El primer paso a realizar en este método es leer el archivo de texto plano subido a la plataforma con el método `read_txt_file`, que guarda cada uno de los *tags* que el usuario quiere conservar en el documento final con su correspondiente comando LaTeX.

```
class Translator
...
  def read_txt_format(file)
    lines = file.download.split("\n")
    datos = {}
    lines.each do |line|
      nombre, edad = line.chomp.split(':')
      datos[nombre] = edad
    end
    datos
  end
...
```

Figura 31: Read TXT Format, Translator Class
Fuente: Elaboración propia.

Luego de haber creado el diccionario con las traducciones, procederemos a convertir directamente cada XML subido, convirtiendo cada etiqueta o *tag* del archivo a su correspondiente comando de LaTeX utilizando como apoyo el diccionario creado cuando se leyó el archivo de texto plano.

```
1  class Translator
2    ...
3    def translate_xml_to_latex(xml_doc, datos)
4      root_tag = xml_doc.xpath('//dump')
5      tags_hijos = root_tag.children
6      xml_string = tags_hijos.to_xml
7
8      datos.each do |llave, valor|
9        regex_ruby = /<\s{0,}#{llave}[^>]*>/
10
11       if valor.include?('$')
12         regex_money = /(<\s{0,}#{llave}[^>]*>[^<>]*)<\/\s{0,}#{llave}[^>]*>/
13         replace_tag = xml_string.scan(regex_money)
14         tag_value = tags_hijos.css(llave)
15
16         replace_tag.each do |elemento|
17           valor_aux = valor
18           new_value = valor_aux.gsub('$', tag_value[0].content)
19           xml_string = xml_string.gsub(elemento.first, new_value)
20         end
21         next
22
23       elsif valor.include?('#')
24         elements = xml_doc.xpath("//#{llave}")
25         replace_tag = xml_string.scan(regex_ruby)
26         n = 0
27
28         elements.each do |element|
29           attribute_value = element.attribute('id').value
30           replace_id = valor.gsub(/#[a-zA-Z0-9_]+/, attribute_value)
31           xml_string = xml_string.gsub(replace_tag[n], replace_id)
32           n += 1
33         end
34         next
35       end
36
37       replace_tag = xml_string.scan(regex_ruby)
38
39       replace_tag.each do |element|
40         xml_string = xml_string.gsub(element, valor)
41       end
42     end
43
44     xml_string.gsub('\n', "\n")
45   end
46   ...
47 end
```

Figura 32: Translation XML To Latex, Translator Class
Fuente: Elaboración propia.

Para este método explicaremos de una forma más detallada cada paso realizado:

1. Lo primero que se hará es obtener el tag principal junto con sus nodos hijos convirtiéndolos a String, específicamente entre las líneas 5 y 7.

2. Luego, recorremos el diccionario creado para traducir cada una de las etiquetas guardadas con su respectivo comando; aquí se utilizará Regex.

3. Después, lo que se realizará es que al tener un símbolo $ es porque cada etiqueta que tenga un valor dado deberá representar ese valor en el comando LaTeX, es decir, si tenemos la etiqueta `<name>Eric</name>`, el contenido que para este caso es **Eric** va a ser guardado y mostrado finalmente en el archivo.

4. Para las siguientes líneas 23 y 35, el proceso que se iniciará es verificar si existe un atributo dentro de las etiquetas que necesite ser rescatado para ser posteriormente visualizado en el documento. Es decir, si tenemos un *Tag* de la forma

`<book name= 'Ruby On Rails Introduction' >`, y resulta que para el archivo de traducción guardado tenemos un símbolo **#** junto con el nombre a rescatar **name**, obtendremos el nombre (Ruby On Rails Introduction) y se reemplazará dentro del archivo principal.

5. Como último paso, se procederá a encontrar todas las etiquetas del archivo XML relacionadas al elemento del diccionario de datos que se esté analizando, procediendo a reemplazar cada una de las etiquetas a reemplazar por las etiquetas ya traducidas.

Finalmente, cuando se hayan completado cada uno de los *Tags* a traducir informados por el usuario, se procederá a darle vida a todos los símbolos saltos de línea como `\n` que se encuentren en el archivo para que puedan ser formateados correctamente.

Una vez completada con éxito la traducción, procederemos a limpiar el código generado para evitar contener etiquetas no deseadas en el documento final con el método `clear_latex`:

```
class Translator
  ...
  def clear_latex(latex)
    regex_complete_clean = %r{<[A-Za-z]*[^<>]*>[^<>\\]*</[^<>]*>}
    regex_tag_clean = /<[A-Za-z]*[^<>\\]*>/
    replace_tag = latex.scan(regex_complete_clean)
    replace_tag.each do |elemento|
      latex = latex.gsub(elemento, '')
    end
    replace_tag = latex.scan(regex_tag_clean)
    replace_tag.each do |elemento|
      latex = latex.gsub(elemento, '')
    end
    latex
  end
  ...
end
```

Figura 33: Cleaning XML, Translator Class
Fuente: Elaboración propia.

Como el software ya está en la parte final del proceso de conversión o transformación de la estructura de la información, todo lo que quede y esté relacionado con etiquetas XML significa que no se va a utilizar para el informe final, debido a que el usuario nunca informó sobre cuáles serían los comandos que posteriormente serían traducidos para dichas etiquetas restantes. Por lo tanto, solamente nos quedan *Tags* "basura"que deben ser eliminadas, ya que estorban en el documento final.

Cuando se haya completado la limpieza se procederá a crear el archivo con formato tex del XML que se ha tomado para ser transformado con el método

`create_xml_translated_tex_file`:

```
class Translator
  ...
  def create_xml_translated_tex_file(file, latex_code)
    FileUtils.chdir template_dirname do
      File.open("#{file.filename}.tex", 'w') { |file| file.write(
    latex_code) }
    end
  end
  ...
end
```

Figura 34: Create Tex File, Translator Class
Fuente: Elaboración propia.

Finalmente, el último proceso a desarrollar corresponde a la compilación del archivo PDF final y la compresión del archivo ZIP que serán guardados en el disco, para posteriormente ser entregados al usuario. En caso de que todo este proceso falle, se hará la captura correspondiente del error y será mostrado al usuario:

```ruby
class Translator
  ...
  def compile_to_pdf_and_compress_zip_file
    output = nil
    pdf_path = nil
    FileUtils.chdir template_dirname do
      output = Open3.capture2e('pdflatex', '-halt-on-error', 'main.tex')
      pdf_path = File.join(template_dirname, 'main.pdf')
    end

    if output.last.exitstatus.zero?
      FileUtils.chdir File.dirname(File.dirname(pdf_path)) do
        system('zip', '-r', "#{File.basename(File.dirname(pdf_path))}.zip
    ", File.basename(File.dirname(pdf_path)))
        @translated_file_path = pdf_path
        @translated_zip_file_path = File.join(File.dirname(File.dirname(
    pdf_path)), "#{File.basename(File.dirname(pdf_path))}.zip")
        true
      end
    else
      regex_scan_error = /!(.*)!  ==> (.*!)/m
      # regex_scan_error = /!(.*)!  ==>/m
      error = output.first.scan(regex_scan_error)
      @error_code = error.first.second
      @error_message = error.first.first
      false
    end
  end
  ...
end
```

Figura 35: Create PDF and ZIP, Translator Class
Fuente: Elaboración propia.

3.10. Tiempo de procesamiento

Para calcular el tiempo de procesamiento de nuestra plataforma finalizada, verificaremos el archivo principal, que puede ser el objeto con mayores cambios durante la subida de archivos, correspondiente al archivo XML.

La razón de esto es que es el único archivo que puede contener grandes cantidades de información y, así, ralentizar mucho más el proceso. En contraste, el archivo de traducción, que viene en formato de texto plano, rara vez tendría un proceso tan pesado, ya que solo es de lectura y se encuentra guardado en memoria. Mientras que para los archivos XML existe lectura y, a su vez, escritura para la compilación final del LaTeX.

Otro supuesto que debemos considerar es que el tiempo se calculará desde que se inicia la

ejecución del archivo, es decir, la lectura del TXT y los XML adjuntos, hasta la conversión final a LaTeX y su posterior compilación a PDF. No se tomará en cuenta desde que es subido por el usuario hasta que entra al proceso de cola, ya que podríamos encontrarnos en la situación de que el sistema de cola esté completo o vacío, lo cual no se podría determinar con certeza, ya que esto dependerá exclusivamente de los usuarios que estén utilizándolo en ese momento.

La cantidad de archivos que se utilizarán para medir el tiempo de ejecución será de seis, cada uno con un peso diferente, de manera que podamos observar con mayor detalle cómo va afectando el rendimiento de la aplicación a medida que nuestros archivos crecen. Para ello, se tienen:

- **Archivo 1:** 0.1 MB
- **Archivo 2:** 0.3 MB
- **Archivo 3:** 0.5 MB
- **Archivo 4:** 0.8 MB
- **Archivo 5:** 1.0 MB
- **Archivo 6:** 1.5 MB

Una vez que tengamos los archivos listos para su ejecución, procederemos a subir los documentos a la plataforma uno por uno, manteniendo constante el archivo TXT y el archivo ZIP, dado que su procesamiento en la aplicación no influye significativamente.

fileing

Tabla 6: Tiempo de ejecución de la plataforma
Fuente: Elaboración propia

Peso archivo XML [MB]	Tiempo de ejecución [SEG]
0.1	0.349256
0.3	1.516087
0.5	4.790734
0.8	11.831479
1.0	16.496827
1.5	37.342749

En esta tabla no podemos determinar con precisión cuál es el comportamiento de nuestra plataforma al aumentar el tamaño del archivo en MB, pero sí podemos hacerlo a través de un gráfico.

Observamos que el tiempo de ejecución aumenta a medida que incrementamos el tamaño de nuestro archivo. Sin embargo, también podemos notar que este aumento no es lineal,

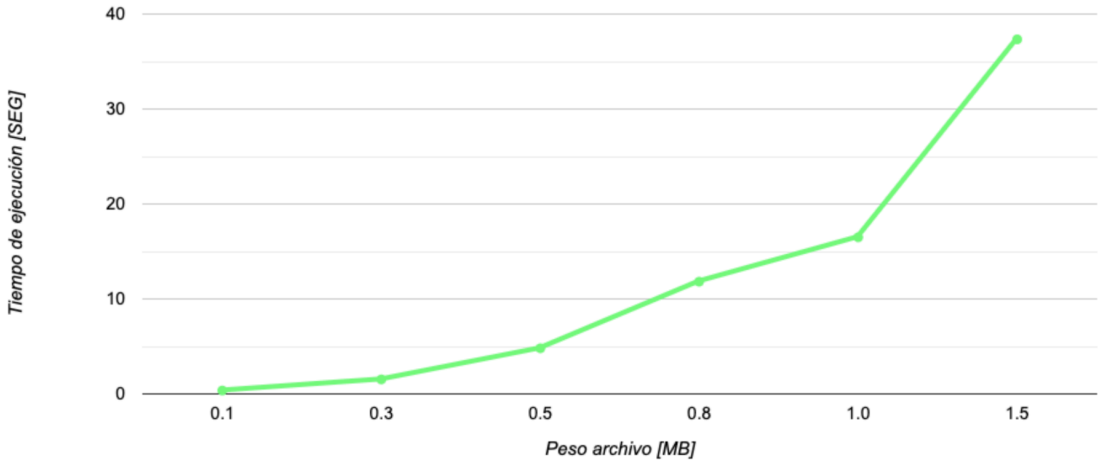


Figura 36: Gráfica de ejecución
Fuente: Elaboración propia

como indica la forma de nuestra gráfica. Esto sugiere que el incremento es de forma exponencial conforme el tamaño del archivo crece.

A continuación, presentamos la ecuación de la recta:

$$y = 2{,}2196 \cdot e^{1{,}8925 \cdot x} \tag{1}$$

Donde vemos que **X** representa la cantidad en MB de los archivos, y **Y** representa la cantidad en segundos aproximados que la aplicación podría tardarse en procesar los archivos.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

4.1. Introducción

La validación de la solución o a lo que nosotros llamaremos *Testing* es uno de los más importantes procesos a la hora de construir un Software. Es un punto que puede marcar bien la diferencia entre un Software de buena y mala calidad, permitiendo ver cuales son las deficiencias y posibles mejoras que podríamos realizar para mejorar su eficiencia. En este capitulo lo que haremos será exclusivamente enfocado en la viabilidad que hemos desarrollado en el capítulo anterior, poniendo también a prueba todo nuestro seguimiento y propuesta realizados para ver que tan sólidas y prácticas fueron.

Lo que haremos en esta sección sera realizar distintas pruebas exhaustivas del producto, es decir, realizar todas aquellas situciones que podría enfrentan nuestro producto y ver si la reacción y respuesta que tiene es la que nosotros esperabamos.

Las pruebas serán divididas en dos secciones, una serán las pruebas unitarias que son aquellas que se prueba individualmente cada función que trabaja en el proceso de conversión o traducción de un archivo desde el principio hasta el final. La segunda sección son las llamadas ”pruebas manuales”que consisten en que uno o varios usuarios de prueba puedan seguir ciertas tareas que contienen conjunto de instrucciones bien definidos a seguir, todo esto se realizará apoyandose a través de la interfaz gráfica del sistema que tiene la plataforma online.

Como último dato y no menos importante en el proceso de *Testing* cada una de las pruebas debe ser solida y bien argumentada utilizando una persepctiva no solo en la validación de una solución, sino también en las posibles falencias o defectos que podrían provocar ciertos ***inputs*** que se ingresen al sistema, esto quiere decir que nos referimos a archivos dañados, que no cumplen un formato correcto, que puedan venir con algún *script* malicioso o que no cumple con el formato establecido tanto para el *XML* como para el archivo de texto a ingresar, entre muchos otros

Con esto dicho anteriormente destacamos la firme importnacia en logro de este proceso para que nuestros resultados pueden ser confiables y durareros a lo largo del tiempo garantizando en su punto máximo la solidez de nuestra propuesta guiando a la implementación en el camino del exito al cúal fue diseñada.

4.2. Pruebas unitarias

En esta sección, lo primero que se nos viene a la mente es realizar pruebas a cada método que hemos creado para verificar su fiabilidad. Pero debemos hacernos una pregunta: ¿Qué es lo que vamos a validar? Son muchos los casos que se nos ocurren, pero algunos en particular son los necesarios para lograr el objetivo de buscar el éxito, que son los llamados çasos borde", donde se desarrollan exclusivamente para los casos donde el software a implementar pudiera tener fallas.

Para ello, dividiremos las pruebas unitarias en 2 secciones, cada una para una clase distinta.

4.2.1. Clase Translation

En esta clase haremos *testing* de los archivos recibidos y el sistema de colas, describiendo cada una de los *tests* para este Software.

1. **Guardar una traducción sin un archivo txt.**

```
1    class TranslationTest < ActiveSupport::TestCase
2    ...
3      test "should not save translation without txt_file" do
4        translation = Translation.new
5
6        assert_not translation.save
7      end
8    ...
9
```

Figura 37: Guardar en BD sin txt
Fuente: Elaboración propia.

Verifica que una instancia de una clase no se guarde en una base de datos si es que no contiene un archivo txt.

2. **Guardar en BD sin archivos XML.**

```
1   class TranslationTest < ActiveSupport::TestCase
2   ...
3     test "should not save translation without xml_files" do
4       translation = Translation.new
5       translation.txt_file.attach(io: File.open("test/fixtures/files/
    test.txt"), filename: "test.txt", content_type: "text/plain")
6
7       assert_not translation.save
8     end
9   ...
10
```

Figura 38: Guardar en BD sin XML
Fuente: Elaboración propia.

El siguiente *test* presentado verifica que no se puedan almacenar los archivos de traducción para su posterior conversión sin el o los archivos XML.

3. **Guardar en BD con archivos txt, XML y Zip.**

```
1   class TranslationTest < ActiveSupport::TestCase
2   ...
3     test "should save translation with txt_file, xml_files and zip_file
    " do
4       translation = new_translation_from_fixtures
5
6       assert translation.save
7     end
8
9     private
10
11    def new_translation_from_fixtures
12      Translation.new do |translation|
13        translation.txt_file.attach(io: File.open("test/fixtures/files/
    test.txt"), filename: "test.txt", content_type: "text/plain")
14        translation.xml_files.attach(io: File.open("test/fixtures/files
    /test.xml"), filename: "test.xml", content_type: "application/xml")
15        translation.xml_files.attach(io: File.open("test/fixtures/files
    /test2.xml"), filename: "test2.xml", content_type: "application/xml")
16        translation.zip_file.attach(io: File.open("test/fixtures/files/
    example.zip"), filename: "example.zip", content_type: "application/zip
    ")
17      end
18    end
19  ...
20
```

Figura 39: Guardar en BD con txt, XML y Zip
Fuente: Elaboración propia.

La siguiente prueba unitaria comprueba si los archivos XML, Zip y txt pueden guardarse en la base de datos si esque están presentes.

4. **Poner en la cola un Job, para posteriormente realizar la traducción del archivo.**

```
      class TranslationTest < ActiveSupport::TestCase
      ...
        test "should enqueue job to process translation" do
          translation = new_translation_from_fixtures

          assert_enqueued_with job: Translation::ProcessingJob do
            translation.save!
          end

          perform_enqueued_jobs

          assert translation.reload.completed?
        end
      ...

```

Figura 40: Encolar un Job
Fuente: Elaboración propia.

Verificamos que si al cargar los archivos necesarios para traducir los documentos a LaTeX, estos se puedan encolar bien para realizar correctamente la conversión.

5. **Verifica que la conversión hecha a LaTeX se elimine después de cinco minutos.**

```
      class TranslationTest < ActiveSupport::TestCase
      ...
        test "should enqueue job to incinerate translation" do
  freeze_time

          assert_enqueued_with job: Translation::IncinerationJob, at:
 5.minutes.from_now do
            create_translation_from_fixtures.process
          end

          travel 5.minutes

          assert_difference -> { Translation.count }, -1 do
            perform_enqueued_jobs only: Translation::IncinerationJob
          end
        end
      ...

```

Figura 41: Encolar un Job
Fuente: Elaboración propia.

Finalmente, como último test unitario para esta clase, verificaremos si la conversión realizada con éxito al formato LaTeX, sea eliminado pasados los cinco minutos predeterminados que hemos definido.

4.2.2. Clase Translator

Para esta clase tendremos un test importante donde veremos si la traducción finalmente ha dado resultados como uno esperaba, es decir, si se ha traducido correctamente el archivo:

```
1       class TranslationTest < ActiveSupport::TestCase
2       ...
3         setup do
4           translation = translations(:first)
5           @translator = Translator.new(txt_file: translation.txt_file,
    xml_files: translation.xml_files)
6         end
7
8         test "verify translation xml" do
9           @translator.translate
10          assert_equal @translator.file_path, translation.
    zip_file_success
11        end
12
13      ...
14
```

Figura 42: Encolar un Job
Fuente: Elaboración propia.

Lo que hace esta clase en este Test es cargar los archivos a través de la clase `Translation`, de manera que esos archivos puedan ser utilizados por la clase `Translator`, todo esto antes de inicializar cualquier Test.

Luego de lo anterior, los archivos de prueba que ya están adjuntos y leídos por la clase `Translation`, realizan la traducción de los archivos y ejecutan el método principal de la clase llamado `translate`, guardando el resultado final del proceso para poder finalmente compararlo con un archivo convertido correctamente, de manera que podamos ver qué tan buena es la eficiencia y calidad de nuestra clase.

4.2.3. Resultados de las pruebas unitarias.

Para ejecutar las pruebas unitarias, Rails posee un comando a través de la terminal que nos permite ejecutar todas las pruebas que fueron realizadas en el framework. A continuación

el comando para ejecutar todas las pruebas a través de la terminal:

```
@MNG-C02FC7W6Q05F lifyx-proyect % rails test
```

`rails test` Ejecuta cada una de las pruebas unitarias realizadas. Después de haber ejecutado el comando de forma correcta, vemos que todos los test han pasado y que las pruebas unitarias demuestran que el código funciona como se esperaría.

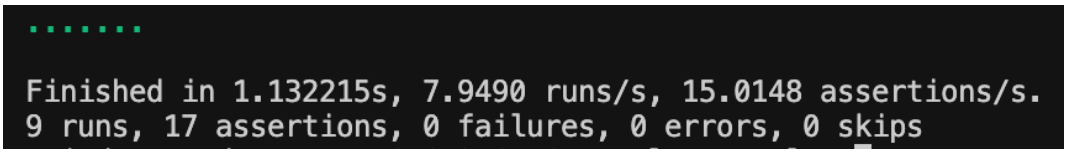


Figura 43: Resultados Tests
Fuente: Elaboración propia.

4.3. Pruebas manuales

Las pruebas manuales en el desarrollo de software son un proceso fundamental para garantizar la calidad de una aplicación o sistema. Estas pruebas son un tipo de evaluación realizada por seres humanos, llamados *testers*, sobre una aplicación de software para identificar posibles errores o problemas.

La razón por la que utilizamos este tipo de pruebas es evaluar la eficacia de la interfaz de usuario en términos de usabilidad y eficiencia en los diferentes entornos en los que se utilizará, simulando también el escenario real que enfrentará cualquier tipo de usuario al acceder a la plataforma.

En esta sección, identificaremos dos ambientes de casos de prueba para probar el entorno de la aplicación web lo más exhaustivamente posible, centrándonos en la funcionalidad del software.

4.3.1. Prueba con múltiples archivos XML

En este tipo de prueba, probaremos la plataforma que hemos desarrollado a través de una interfaz gráfica. Utilizaremos el archivo principal de texto plano, además de dos archivos XML, sin tener una plantilla definida en el archivo ZIP, con la intención de verificar si nuestra plantilla por *default* funciona y si puede traducir varios archivos a la vez. Esta parte del informe será de mucha utilidad.

Para empezar, utilizaremos un archivo de texto plano como el siguiente ejemplo:

```
SEc: \subsection {asdas}
Book: \section {#id} \n \begin{itemize} \n \item{Item 1}
character: \item {\textbf{char}}
Author: \begin{itemize}
name: \item{$}
age: \item{$}
/Author: \end{itemize}
childrens: \item{$}
Title: \item {$$}
Genre: \item {$}
PublishDate: \item {$}
Description: \item {$}
/Book: \end{itemize}
```

Figura 44: Archivo de traducción para XML
Fuente: Elaboración propia.

Veremos que, para robustecer esta prueba, hay algunas etiquetas que no estarán en los archivos XML que se irán a traducir, lo que aumenta el nivel de prueba para el caso manual.

Luego, tendremos los archivos XML. Para el primer caso, tendremos:

```xml
<?xml version="1.0"?>
<dump>
    <Book id="Booking 01">
        <character>
            <Author id="2">
                <name>John Doe</name>
                <age>36</age>
                <childrens>5</childrens>
            </Author>
            <Title>XML Developer's Guide</Title>
            <Genre id="1">Computer</Genre>
            <Price>44.95</Price>
            <PublishDate>2000-10-01</PublishDate>
            <Description>An in-depth look at creating applications with XML.</Description>
        </character>
    </Book>
    <Book id="Boooking 02">
        <character>
            <Author id="2">
               <name>Johnny Clark</name>
               <age>53</age>
               <childrens>5</childrens>
            </Author>
            <Title>Midnight Rain</Title>
            <Genre id="2">Fantasy</Genre>
            <Price>5.95</Price>
            <PublishDate>2000-12-16</PublishDate>
            <Description>A former architect battles corporate zombies, an evil sorceress, and her
                own childhood to become queen of the world.</Description>
        </character>
    </Book>
</dump>
```

Figura 45: Archivo 1 XML
Fuente: Elaboración propia.

Mientras que el segundo archivo XML tendrá el siguiente contenido:

```xml
<?xml version="1.0"?>
<dump>
    <Book id="Booking 01">
        <character>
            <Author id="6">
               <name>Alberth</name>
               <surname>Zuckenberg</surname>
               <age>25</age>
               <childrens>1</childrens>
            </Author>
            <Title>My life</Title>
            <Genre id="76">Love</Genre>
            <Price>44.95</Price>
            <Description>Error! not description found.</Description>
        </character>
    </Book>
</dump>
```

Figura 46: Archivo 2 XML
Fuente: Elaboración propia.

Ahora que tenemos nuestros dos archivos XML listos, junto con el archivo de traducción, procederemos a ir a la plataforma para realizar la transformación.

Figura 47: Página principal de Lifyx
Fuente: Elaboración propia.

Para ello, ingresamos el archivo de traducción donde corresponde y los dos archivos XML en el campo asignado, dejando solo la opción de subida de archivo ZIP vacía. Con esto, probaremos si funciona la conversión que deseamos realizar.

Se hará clic en el botón Çonvertir"para ser dirigidos a una página que indique que se está traduciendo, cumpliendo con el derecho de informar al usuario sobre el estado de la aplicación o proceso en desarrollo.

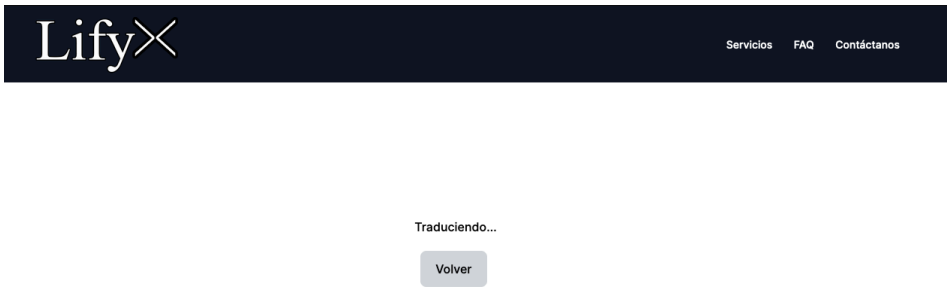


Figura 48: Muestra de estado de conversión
Fuente: Elaboración propia.

Después de que el proceso de conversión haya finalizado, se nos mostró un mensaje exitoso indicándonos que la conversión se ha completado de forma correcta:

Figura 49: Conversión completada con éxito
Fuente: Elaboración propia.

Se nos muestran dos botones, cada uno permitiendo descargar ya sea el archivo PDF o el archivo ZIP generado por nuestra plataforma utilizando la plantilla por defecto. Si descargamos y analizamos el PDF generado, se mostrará lo siguiente:

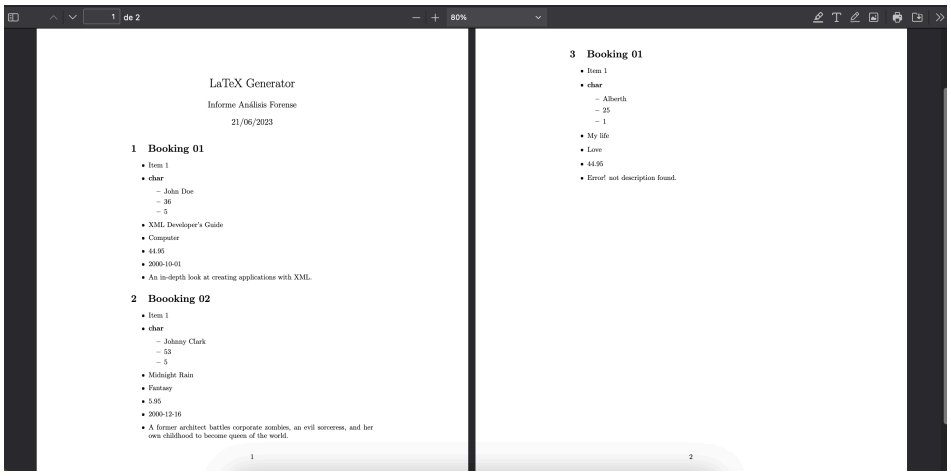


Figura 50: PDF Compilado
Fuente: Elaboración propia.

Esto nos indica que el archivo se ha convertido a PDF de forma correcta. Ahora, debemos asegurarnos de que el archivo ZIP contiene los archivos XML con sus respectivos nombres. Para ello, descargaremos el archivo ZIP con el botón ”Descargar ZIP File, posteriormente, abriremos la carpeta para verificar los archivos.

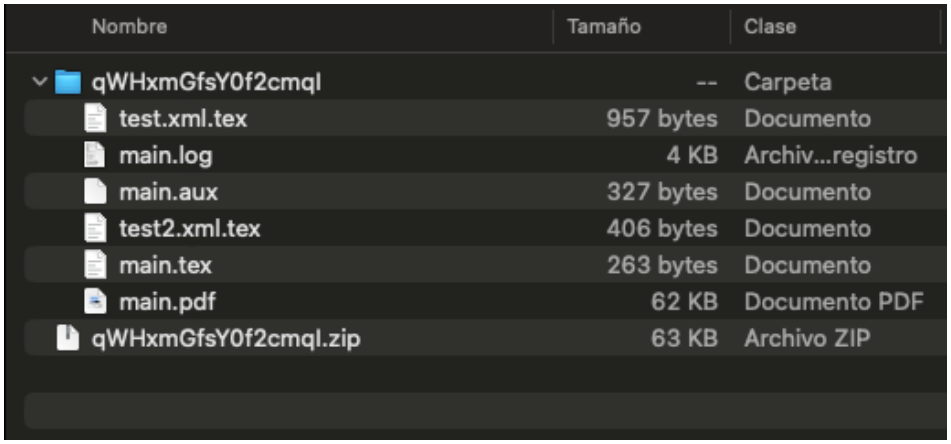


Figura 51: Archivos ZIP descargados
Fuente: Elaboración propia.

Todo salió de maravilla, por lo que podemos decir que la primera prueba manual ha sido un rotundo éxito!

4.3.2. Prueba con múltiples archivos XML y un archivo ZIP

En este último caso de prueba, utilizaremos los mismos archivos de la sección anterior, pero añadiremos una plantilla subida por el usuario. La plantilla comprimida será la siguiente:

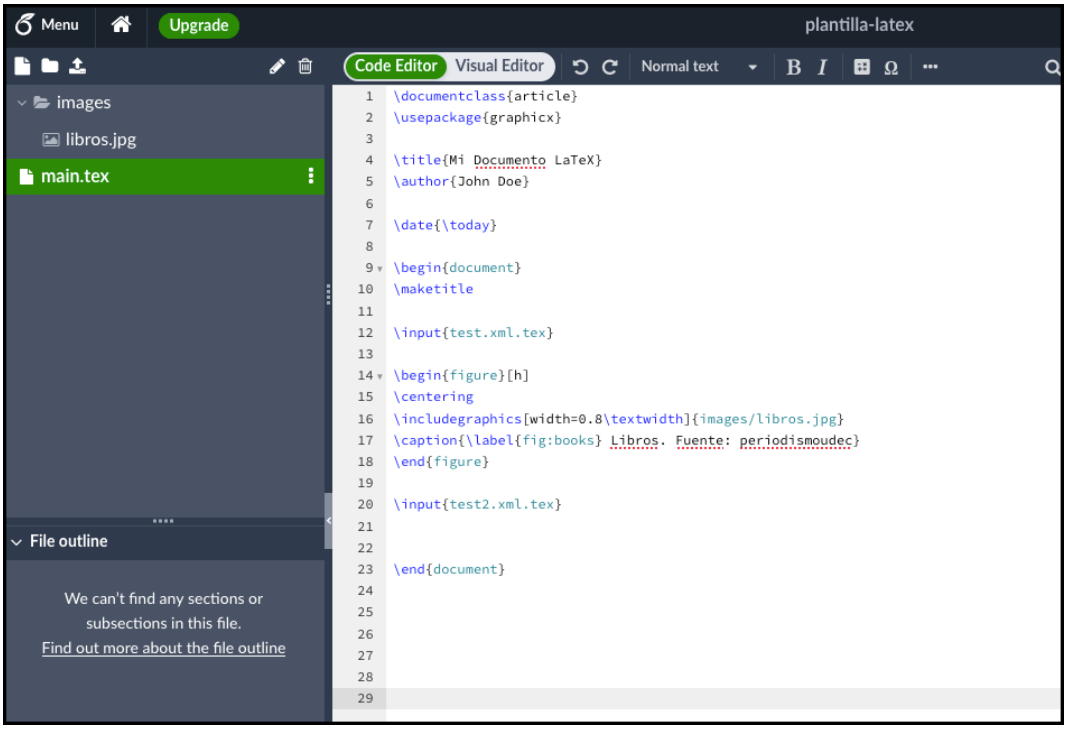


Figura 52: Plantilla ZIP del usuario
Fuente: Elaboración propia.

Podemos apreciar que existe un comando LaTeX `\input` que importa y agrega el contenido de los archivos subidos por el usuario, los cuales fueron convertidos previamente. En nuestro caso de prueba, se trataría de los archivos XML test y test1.

Cargamos cada uno de los archivos en sus campos correspondientes para que la plataforma pueda leerlos correctamente, tal como se muestra en la siguiente imagen:

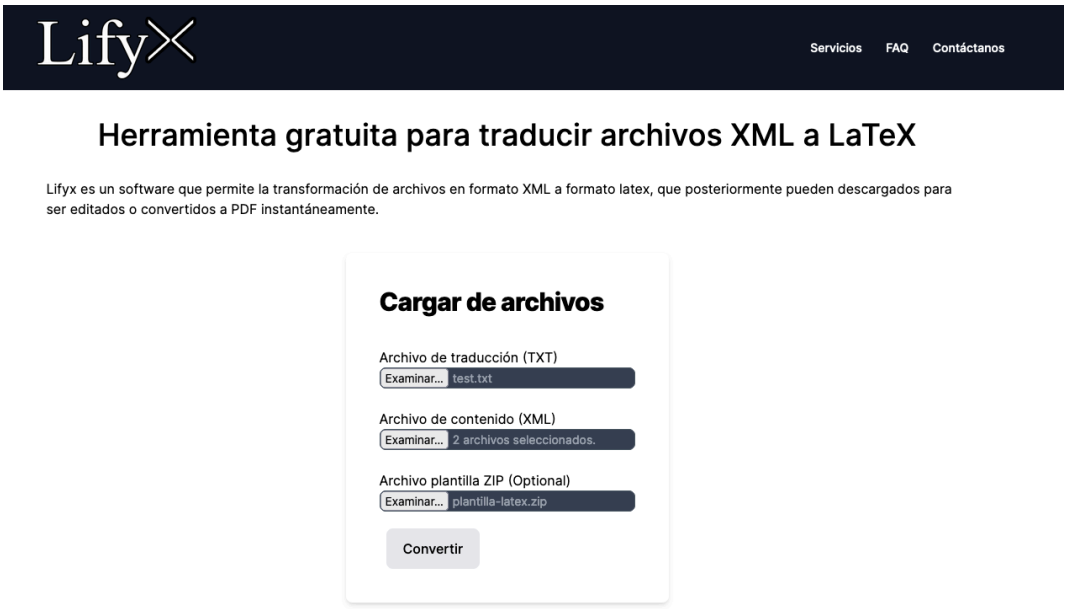


Figura 53: Interfaz de Lifyx con los 3 archivos
Fuente: Elaboración propia.

Al hacer clic en el botón de convertir, se muestra el estado de la ejecución de la aplicación web. De esta manera, independientemente de si se sube o no un archivo ZIP como plantilla, se mantiene la consistencia y la experiencia del usuario al informarle sobre el progreso del proceso.

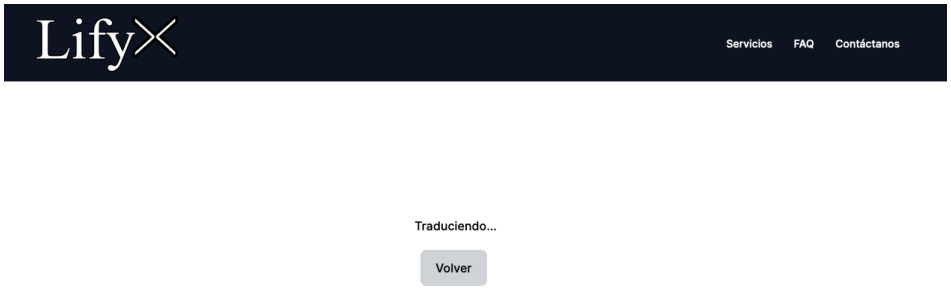


Figura 54: Muestra de estado de conversión utilizando archivo ZIP
Fuente: Elaboración propia.

Y observamos que, de igual manera, se mostrará un mensaje indicando si el proceso se completó de manera exitosa, como es el caso, o no.

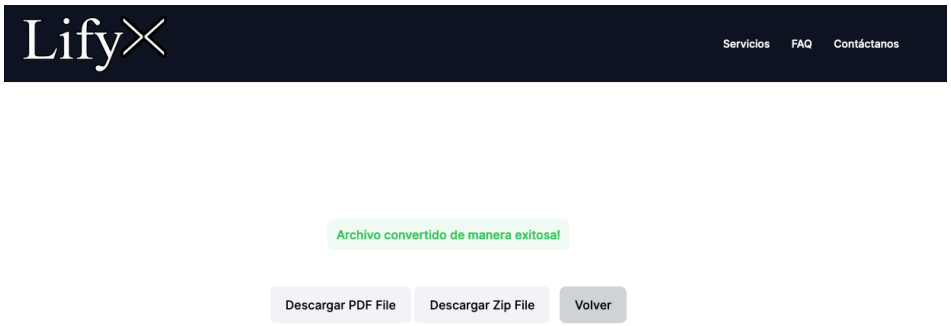


Figura 55: Conversión completada con éxito utilizando archivo ZIP
Fuente: Elaboración propia.

Descargamos el archivo en formato PDF y comprobamos que se muestra tal como se había pronosticado en la plantilla de la figura 44, donde el contenido del primer archivo XML debe agregarse antes de la imagen, mientras que el segundo archivo XML debe agregarse después de la imagen.

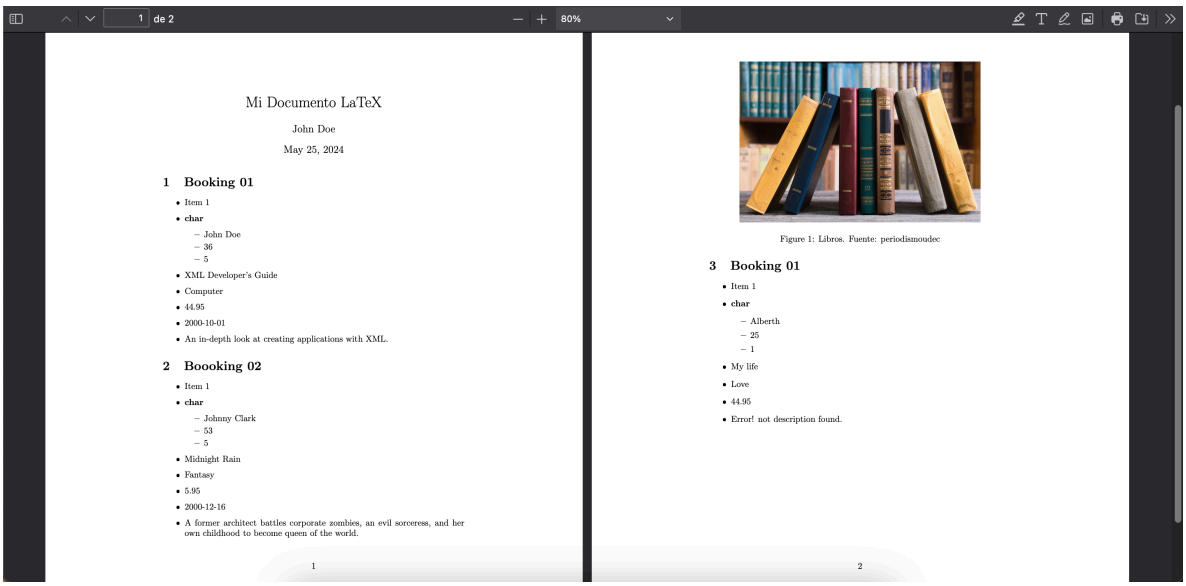


Figura 56: PDF con ZIP del usuario compilado
Fuente: Elaboración propia.

Esto de igual manera debe verse en el documento comprimido.

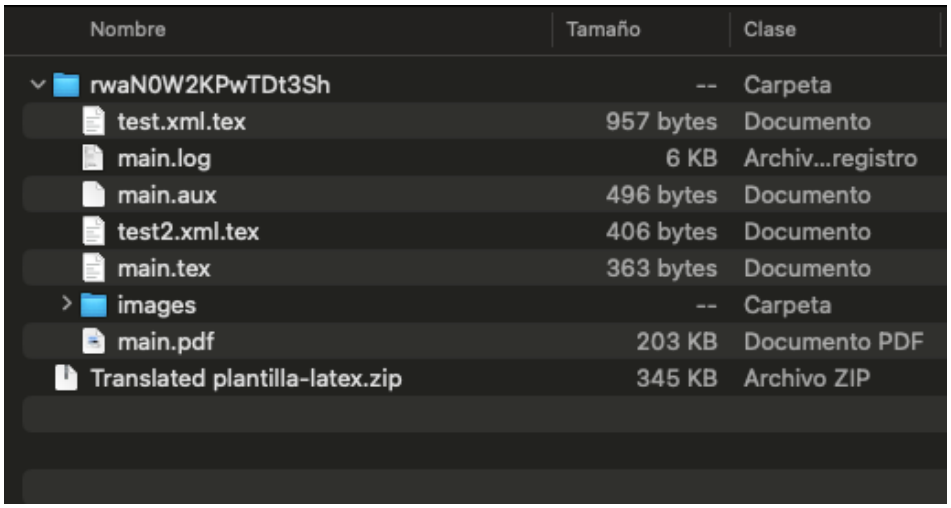


Figura 57: Archivo ZIP del usuario descargado
Fuente: Elaboración propia.

Todo se completó de la forma esperada: los archivos PDF compilados se muestran con la información legible y en la estructura prevista. Además, el archivo comprimido contiene los XML traducidos a formato LaTeX, manteniendo intactos los demás documentos subidos por el usuario.

El desarrollo y los buenos resultados obtenidos en esta sección evidencian que la aplicación web funciona correctamente, tanto para los diversos casos de uso como para los casos extremos que podrían causar fallos en la plataforma. Sin duda, pudimos comprobar que la eficiencia y el rendimiento del software desarrollado destacan gracias al arduo trabajo invertido en su desarrollo.

CAPÍTULO 5
CONCLUSIONES

5.1. Conclusión general

En esta memoria, lo que se estudió fue la transformación de archivos dado un formato en específico, que sería un archivo formato XML para poder ser finalmente transformado a un archivo LaTeX. Se revisaron diversas investigaciones sobre cuál es la mejor forma de poder implementar esta solución y además de cómo implementarla asegurando el trabajo de calidad y limpio que requiere una institución tan importante en Chile como lo es la Policía de Investigaciones, proponiendo un modelo y solución para dicho tema.

La solución se diseñó implementando una plataforma web donde el usuario deberá subir tres tipos de archivos distintos: un archivo de traducción, el archivo con la información principal en XML y un archivo ZIP opcional en caso de que ya se tenga una plantilla definida para poder mostrar la información.

La plataforma se logró utilizando un Framework muy actual denominado Ruby on Rails que se enfoca en la simplicidad y eficiencia del programador para poder desarrollar soluciones robustas y rápidas. Además de la implementación de una base de datos para almacenar información precisa que pueda servir a futuro y un servidor para el alojamiento de la aplicación.

También se logró desarrollar nuestra propuesta de solución utilizando regex y programación orientada a objetos (POO), donde la primera se basa en definir ciertos patrones para encontrar las etiquetas necesarias que se requieren ya sea para traducir un tag o bien para eliminar si es que el usuario no lo ha definido en el archivo de traducción, mientras que la POO fue utilizada para mantener código limpio y ordenado utilizando buenas prácticas como los nombres de los métodos y la cantidad de información que procesará cada uno.

Nuestro modelo propuesto fue validado por diferentes casos de pruebas realizando pruebas unitarias, manuales y de la misma institución, donde cada una de ellas se realizó utilizando casos límite para mejorar la calidad de nuestro software y su refinamiento en caso de que alguna de ellas falle. Permitiendo así que nuestra plataforma o aplicación web estuviera finalizada con éxito.

Se espera que esta aplicación sea utilizada por la PDI para poder satisfacer sus necesidades y problemas de legibilidad a la hora de leer documentos y que sea implementado en todos los archivos que requieran ser presentados como documentos oficiales. También se espera que a futuro con los datos almacenados se pudiera realizar un tipo de análisis y ver cuáles son los puntos de mejora en nuestro software, de manera que pueda mejorar la calidad de trabajo de los usuarios que lo utilicen.

5.2. Cumplimiento de los objetivos

El cumplimiento de los objetivos es un buen instrumento para medir el éxito y el resultado de la memoria realizada, ya que en su conjunto determinan finalmente el objetivo general. Por lo tanto, se presentará cada objetivo y se evaluará su verdadero éxito:

- **Presentar y analizar el problema en la lectura de archivos generados al finalizar un análisis forense:** En la sección 1.1, pusimos en contexto la situación que tenía la institución de poder presentar esa información en un caso formal y las dificultades que se tenían para poder alcanzar su objetivo de mejorar la legibilidad del documento manteniendo la consistencia de la información. Analizamos incluso cada una de las herramientas al alcance que podrían ser parte de la solución, pero que finalmente no aportaban nada, y menos a la necesidad final del usuario de poder convertir un archivo correctamente. Por lo tanto, consideramos este objetivo como logrado.
- **Definir patrones que relacionen etiquetas a comandos específicos en LaTeX:** Para ello, al principio se tenía la idea de definir patrones por defecto de manera que el usuario solo tuviera que subir el archivo y ver cómo se convertía de manera automática. Esto, en principio, fue un éxito, pero sin embargo, al analizar el caso con mayor detalle, se prefirió elegir la opción de que el usuario pudiera subir la traducción de comandos de cada etiqueta a LaTeX, lo cual, según la sección 4, fue implementado correctamente, pudiendo así completar este objetivo.
- **Diseñar un algoritmo considerando los patrones definidos:** Sin duda, se tiene que admitir que este fue uno de los objetivos más difíciles de realizar, ya que había que tener en cuenta cómo resolverlo, es decir, el proceso del algoritmo en sí explicado en la sección 3.7 y la forma en que se resolvería, como fue en este caso utilizando expresiones regulares y programación orientada a objetos. El algoritmo fue probado en diferentes casos de prueba, incluyendo la institución, resultando con éxito, por lo que este objetivo estaría cumplido.
- **Evaluar el prototipo considerando los diversos casos, emitidos directamente desde la institución:** En la sección anterior, específicamente en la sección 4.3, se realizaron casos de prueba emitidos por parte de la institución de la PDI, donde cada uno de los casos emitidos pudo ser procesado con éxito, finalizando así el objetivo.

Cada uno de los objetivos especificos analizados anteriormente fueron completados con éxito por lo que, ahora lo que debemos pensar es en nuestro objetivo general que corresponde a: **Implementar una plataforma web para transformar archivos de análisis forenses con formato XML a formato Latex.** ¿Realmente se implemento una plataforma?, pues si esta en la web y funcionando 24/7, realmente transforma los archivos de acuerdo a las necesidades del propio usuario, según nuestras validaciones en la sección anterior, en base a esto podemos concluir que todos los objetivos fueron cumplidos.

5.3. Trabajo a futuro

A pesar de que el modelo de la solución se hizo principalmente pensado en la Policía de Investigaciones, sería ideal poder extenderlo a otros tipos de usuarios, ya que es muy común que varias empresas o instituciones trabajen con este tipo de archivo, ya sea porque siempre lo han hecho desde el principio de la era de la informática, donde era muy común trabajar con este tipo de archivos, o simplemente por el propósito para el que fue creado.

Este tema puede ser un muy buen argumento sólido para poder generar un emprendimiento y ayudar a toda entidad que requiera procesar este tipo de información en un archivo que sea legible para distintos objetivos o necesidades que se requieran, dependiendo del contexto en que se encuentre.

Sin embargo, se requiere una investigación muy profunda, ya que las necesidades de cada organización pueden ir variando y no podrían ser muy equivalentes unas necesidades con otras. Es imposible saber, sin interactuar con el usuario, qué es lo que realmente quiere, por lo que esto podría considerarse un trabajo de título debido a la complejidad que tiene generar una solución genérica a nivel universal, adaptándose a cada usuario que utilice la plataforma.

Otro posible trabajo de título sería una implementación con algoritmos de Machine Learning que permita identificar y traducir de manera automática todos los tags XML que se encuentren en el archivo, que pueda identificar de acuerdo a un historial del usuario o parámetros que sean enviados por este para lograr descargar un archivo de forma correcta, donde vemos claramente que aquí hay una complejidad que se debe profundizar realizando investigaciones.

Recordemos también que en la sociedad cada vez se está haciendo más habitual y oficial el uso de dispositivos móviles para todo, ya sea pagos, mensajería, llamadas, etc. Por lo que también el desarrollo de una aplicación móvil sería una de las tantas buenas ampliaciones que se le podrían realizar a este proyecto, pudiendo llevar esta aplicación a cualquier parte donde se encuentre el usuario sin la necesidad de limitaciones como, por ejemplo, la de tener sí o sí un computador.

5.4. Palabras finales del autor

Realizar esta memoria fue un desafío muy interesante de enfrentar y desarrollar, ya que desde un principio teníamos en conocimiento el problema a desarrollar, entendiendo primero cuál era la necesidad y el objetivo al que se quería llegar, pero no se tenía idea de cómo proponer una buena solución debido a que podría ser abordado de diferentes formas y cada una tenía sus ventajas y desventajas. Por lo que hubo que hacer una investigación bien profunda y clara de cómo desarrollarla en base a las necesidades del cliente y cómo abordarla para que pudiera mantenerse durante el tiempo.

Notamos que un software implementado como aplicación de escritorio o como aplicación web tiene muchas diferencias significativas en base a su mantenibilidad y/o escalabilidad, aparte de la tecnología usada de acuerdo a la solución propuesta. Aprendimos también que la conversión y transformación de archivos puede ser compleja si es que no se tiene clara la verdadera necesidad del usuario.

Es por ello que me es muy gratificante haber logrado lo que me propuse, ya que es el primer paso profesional que logré hacer basándome en la enseñanza que tuve en toda mi carrera, el transformar la información de un tipo de archivos a otro asegurando la consistencia de la información y la necesidad del usuario me da un potencial para poder seguir abarcando este y otro tipo de problemas, mejorando la experiencia de las personas o incluso organizaciones, satisfaciendo las necesidades a través del desarrollo de software.

REFERENCIAS BIBLIOGRÁFICAS

[Abogados, 2022] Abogados (2022). El delito de acceso ilicito a los sitemas informáticos. *Castella Abogados*. https://www.castellabogados.com/el-delito-de-acceso-ilicito-a-los-sistemas-informaticos/.

[Adobe, 2022] Adobe (2022). What is pdf. *Adobe Reader*. https://www.adobe.com/acrobat/about-adobe-pdf.html.

[Amazon, 2022] Amazon (2022). What is xml. *Amazon*. https://aws.amazon.com/es/what-is/xml/#:~:text=El%20lenguaje%20de%20marcado%20extensible,datos%20y%20aplicaciones%20de%20terceros.

[Amazon, 2023] Amazon (2023). ¿qué es la ciencia de datos? *Amazon*. https://aws.amazon.com/es/what-is/data-science/#:~:text=L.

[Anjelino, 2022] Anjelino (2022). ¿qué características tiene un documento en formato pdf? *harasdadinco*. https://www.harasdadinco.cl/que-caracteristicas-tiene-un-documento-en-formato-pdf/.

[ASpose, 2022] ASpose (2022). Xml to latex. *Convertion*. https://products.aspose.app/pdf/es/conversion/latex-to-xml.

[Baumann, 2021] Baumann, H. (2021). Ruby, el lenguaje de programación elegante y potente que debes aprender. *Crehana*. https://www.crehana.com/blog/transformacion-digital/ruby-lenguaje-programacion/.

[BCN, 2022] BCN (2022). Delitos informáticos. *Biblioteca del congreso*. https://www.bcn.cl/portal/leyfacil/recurso/delitos-informaticos.

[CampusMVP, 2023] CampusMVP (2023). Por qué java sigue siendo el lenguaje número 1. *CampusMVP*. https://www.campusmvp.es/recursos/post/por-que-java-sigue-siendo-el-lenguaje-numero-1.aspx#::text=En.

[Chakray, 2022] Chakray (2022). Lenguajes de programación: tipos y características. *Chakray*. https://www.chakray.com/es/lenguajes-programacion-tipos-caracteristicas/.

[Daniel-Penaloza, 2021] Daniel-Penaloza (2021). Bloques con ruby. *Dev*. https://dev.to/danielpenaloza/bloques-con-ruby-1ai0.

[DropBox, 2022] DropBox (2022). What is a digitization. *DropBox experience*. https://experience.dropbox.com/es-la/resources/what-is-digitization.

[Eseme, 2023] Eseme, S. (2023). Los 10 tipos de aplicaciones node.js más populares en 2024. *Kinsta*. https://kinsta.com/es/blog/node-js-aplicaciones/.

[Frisoli, 2023] Frisoli, C. (2023). Los 12 mejores frameworks para desarrollo web en 2024. *Hubspot*. https://blog.hubspot.es/website/framework-desarrollo-web.

[Gobierno, 2022] Gobierno (2022). Nueva ley de delitos informaticos entra en vigor. *CSIRT, Gobierno de chile*. https://ciberseguridad.gob.cl/noticias/nueva-ley-de-delitos-informaticos-entro-en-vigor/.

[Golang, 2022] Golang (2022). Middleware. *Golang*. https://pkg.go.dev/github.com/code-epic/middleware#section-readme.

[Hansson, 2023] Hansson, D. H. (2023). La doctrina rails. *RubyonRails*. https://rubyonrails.org/doctrine/es#convention-over-configuration.

[Holcome, 2022] Holcome, J. (2022). Xml vs html. *Kinsta*. https://kinsta.com/es/blog/xml-vs-html/#:~:text=El%20c%C3%B3digo%20HTML%20est%C3%A1%20hecho,visto%20en%20el%20front%2Dend.

[HostingPlus, 2021] HostingPlus (2021). Ruby: el mejor lenguaje de programación para principiantes. *HostingPlus*. https://www.hostingplus.cl/blog/ruby-el-mejor-lenguaje-de-programacion-para-principiantes/.

[Huet, 2023] Huet, P. (2023). Frameworks java para un desarrollo eficiente. *OpenWebinars*. https://openwebinars.net/blog/frameworks-java-para-un-desarrollo-eficiente/.

[IBM, 2023] IBM (2023). El compilador jit. *IBM*. https://www.ibm.com/docs/es/sdk-java-technology/8?topic=reference-jit-compiler.

[Mira, 2020] Mira, A. R. (2020). ¿programar en java es difícil? conviértete en programador. *TokioSchool*. https://www.tokioschool.com/noticias/programar-en-java-es-dificil/.

[mytaskpanel, 2023] mytaskpanel (2023). Lenguaje de programación ruby: características y utilidades. *mytaskpanel*. https://www.mytaskpanel.com/lenguaje-de-programacion-ruby/.

[Oliver y Mayer, 2020] Oliver, G. y Mayer, L. (2020). El delito de fraude informático: concepto y delimitación. *RCHDT*. https://rchdt.uchile.cl/index.php/RCHDT/article/view/57149/61949.

[Oracle, 2023] Oracle (2023). Evaluación del rendimiento de las bases de datos de un vistazo. *Oracle*. https://docs.oracle.com/es-ww/iaas/database-management/doc/assess-performance-your-databases-glance.html.

[Paramio, 2011] Paramio, C. (2011). Extendiendo la funcionalidad de las librerías básicas de ruby con facets. *Genbeta*. https://www.genbeta.com/desarrollo/extendiendo-la-funcionalidad-de-las-librerias-basicas-de-ruby-con-facets.

[PDI, 2022] PDI (2022). Radiografía de homicidios a lo que van año. *PDI Chile 2022*. https://www.pdichile.cl/centro-de-prensa/detalle-prensa/2022/07/13/radiograf%C3%ADa-a-los-homicidios-en-lo-que-va-de-2022.

[Prakmatic,] Prakmatic. Que es el análisis forense. *Prakmatic*. https://www.prakmatic.com/que-es-el-analisis-forense-informatico/.

[Pérez, 2021] Pérez, S. D. (2021). ¿qué es microsoft sql server y para qué sirve? *intelequia*. https://intelequia.com/es/blog/post/qu%C3%A9-es-microsoft-sql-server-y-para-qu%C3%A9-sirve.

[Romanos, 2023] Romanos, M. (2023). ¿qué es un orm? *Dreams*. https://www.dreams.es/transformacion-digital/desarrolladores-paginas-web/que-es-un-orm.

[Roomi, 2022] Roomi, M. (2022). 6 advantages and disadvantages of spring boot | limitations benefits of spring boot. *HitechWhizz*. https://www.hitechwhizz.com/2022/08/6-advantages-and-disadvantages-limitations-benefits-of-spring-boot.html.

[Ruby, 2023] Ruby (2023). Comunidad. *Ruby-Lang*. https://www.ruby-lang.org/es/community/.

[SonTips, 2024] SonTips (2024). Cómo elegir la mejor laptop del 2024, según tus necesidades. *SonTips*. https://son.tips/finanzas/elegir-mejor-laptop-caracteristicas/.

[TI, 2022] TI (2022). Integridad de datos. *Tecnologías información*. https://www.tecnologias-informacion.com/integridaddatos.html.

[Unir, 2022] Unir (2022). Framework: qué es, para qué sirve y algunos ejemplos. *Unir, tecnología e ingeniería*. https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/.

[universidadviu, 2018] universidadviu (2018). Lenguaje de bajo nivel, características y funciones. *universidadviu*. https://www.universidadviu.com/int/actualidad/nuestros-expertos/lenguaje-de-bajo-nivel-caracteristicas-y-funciones.

[Uniwebsidad, 2023] Uniwebsidad (2023). El patrón de diseño mtv. *Uniwebsidad*. https://uniwebsidad.com/libros/django-1-0/capitulo-5/el-patron-de-diseno-mtv.

[Vailshery, 2023] Vailshery, L. S. (2023). Most used web frameworks among developers worldwide, as of 2023. *Statista*. https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/.

[Villa, 2022] Villa, A. A. (2022). Qué es go y qué usos tiene. *Profile*. https://profile.es/blog/que-es-go-y-que-usos-tiene/.

[Web, 2023] Web, M. (2023). Características javascript. *Manual Web*. https://www.manualweb.net/javascript/caracteristicas-javascript/.

[Wikidot, 2022] Wikidot (2022). Duck typing. *Wikidot*. http://rubytutorial.wikidot.com/duck].

[Wikipedia, 2023] Wikipedia (2023). Python. *Wikipedia*. https://es.wikipedia.org/wiki/Python.