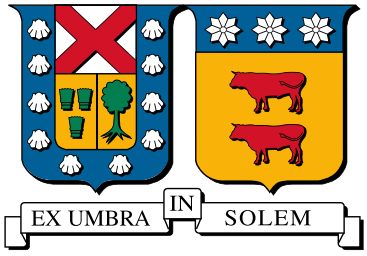
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE



"HERRAMIENTA PARA LA EXTRACCIÓN, PROCESAMIENTO, ANÁLISIS Y VISUALIZACIÓN DE DATOS DE SISTEMAS DE CONTROL AVANZADO"

LUCAS AGUSTÍN PINO JARA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO.

PROFESOR GUIA: DR. WERNER CREIXELL

PROFESOR CORREFERENTE: GUSTAVO BITTNER

ENERO 2024

DEPARTAMENTO DE ELECTRONICA
UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

DEPARTAMENTO DE ELECTRONICA

Herramienta para la extracción, procesamiento, análisis y visualización de datos de sistemas de control avanzado

Lucas Agustín Pino Jara

Memoria para optar al título de Ingeniero Civil Electrónico, mención Computadores, submención Informática.

Universidad Técnica Federico Santa María

Profesor Guía: Dr. Werner Creixell

Profesor Correferente: Gustavo Bittner

ENERO 2024

Resumen

Este trabajo presenta el desarrollo de una herramienta de software para la extracción, procesamiento, análisis y visualización de datos en sistemas de control avanzado. El proyecto tiene como objetivo principal mejorar la eficiencia en el manejo de datos en la industria minera. La solución propuesta se basa en el uso de Dash/Plotly destacándose por su adaptabilidad y funcionalidades interactivas. El documento detalla desde la revisión del estado del arte y los requerimientos del usuario hasta la arquitectura, diseño de la interfaz y tecnologías utilizadas. Incluye pruebas de usabilidad y estrés para validar la herramienta. Las conclusiones resaltan la contribución del proyecto a la estandarización y eficiencia en el análisis de datos en Kairos Mining/Honeywell Chile.

DEPARTAMENTO DE ELECTRONICA
UNIVERSIDAD TECNICA FEDERICO SANTA MARIA

DEPARTAMENTO DE ELECTRONICA

Tool for Extraction, Preprocessing, Analysis, and Visualization of Advanced Control System Data

Lucas Agustín Pino Jara

Thesis for the fulfillment of the B.S. in Electronic Engineering, major in Computer Electronics, minor in computer science (6 year program).

Universidad Técnica Federico Santa María

Advisor: Dr. Werner Creixell

Co-Advisor: Gustavo Bittner

ENERO 2024

Abstract

This thesis presents the development of a software tool for data extraction, processing, analysis, and visualization in advanced control systems. Aimed at improving data management efficiency in the mining industry, the project focuses on a solution utilizing Dash/Plotly, notable for its adaptability and interactive features. The document thoroughly outlines the state-of-the-art review, user requirements, system architecture, interface design, and utilized technologies. It includes usability and stress tests to validate the tool. The conclusions highlight the project's contribution towards standardizing and enhancing data analysis efficiency at Kairos Mining/Honeywell Chile.

Glosario

API Application Programming Interface.

Back-end Se refieren a la lógica de procesamiento de datos.

Callbacks Funciones que responden a eventos del usuario en aplicaciones Dash.

DataFrame Objeto de datos bidimensional en pandas que se utiliza para almacenar y manipular datos tabulares.

DLL Biblioteca de Vínculos Dinámicos (Dynamic Link Library).

Drag and Drop Técnica de interfaz para arrastrar y soltar elementos en la GUI.

Framework Entorno de desarrollo que proporciona estructuras y herramientas para facilitar la creación y el desarrollo de software.

Front-end Término que se refieren a la lógica de procesamiento de datos y la interfaz de usuario, respectivamente.

GUI Interfaz Gráfica de usuario.

Layout Disposición o estructura de elementos en una página o interfaz, especialmente en diseño web y gráfico..

Markdown Lenguaje de marcado ligero para formatear texto.

Timestamp Marca de tiempo que indica cuándo ocurrió un evento específico en un sistema.

Web App Aplicación que funciona a través de un navegador web.

Índice de contenidos

Índice de figuras

1. Introducción

El análisis y la visualización de datos desempeñan un papel crucial en el ámbito del control avanzado, siendo tareas recurrentes y fundamentales para la toma de decisiones informadas y la mejora continua de muchos procesos productivos. La relevancia de estas tareas se acentúa en sectores como la industria minera, un ámbito caracterizado por su intensiva inversión en activos fijos para mejorar constantemente la producción. En esta industria, la maquinaria empleada no solo es pesada, grande y costosa, sino que también es esencial para la rentabilidad del proceso minero.

Por lo tanto, asegurar su funcionamiento óptimo, mantenimiento adecuado y detectar oportunidades de mejora se convierte en una prioridad diaria. La capacidad de extraer, visualizar y analizar datos ayuda a prevenir detenciones no planificadas y prolongar la vida útil de los equipos, al mismo tiempo que facilita la toma de decisiones estratégicas para mejorar la eficiencia y reducir los costos de operación. La monitorización precisa y el análisis de los datos de sistemas de control permiten identificar patrones, predecir fallas potenciales y adoptar medidas preventivas, minimizando así los riesgos para los trabajadores y la organización.

Actualmente los reportes de análisis de datos son creados utilizando múltiples herramientas: una herramienta para importar/extraer los datos de la fuente, otra herramienta para procesar los datos, otra herramienta para generar visualizaciones, y finalmente alguna herramienta como MS Paint[2] para hacer dibujos y anotaciones en las visualizaciones. Aunque hay herramientas muy capaces para todas estas tareas, no hay un consenso sobre qué se debe usar para cada tarea, y cada trabajador es libre de elegir una herramienta por separado para cada paso del proceso; esto lleva a una descoordinación en la gestión de la información y en el lenguaje visual de los reportes dentro de la organización.

1.1. Estado del arte

En el mercado existen soluciones que suplen las necesidades de la industria minera, proporcionando captura de datos, procesamiento de datos, visualizaciones y reportes. El problema es que no todas se adaptan a las necesidades específicas de cada cliente. Esto se vuelve un tema sensible cuando la prioridad del cliente es ser dueño de sus datos, lo cual es un tema recurrente en la industria minera, ya que son datos confidenciales e indispensables, y una fuga de datos o una pérdida parcial debido a la falla de un servidor del proveedor de servicios puede tener consecuencias catastróficas. Por ello, el riesgo adicional de confiar tus datos a un tercero no vale el tiempo ganado en la optimización de los reportes de datos. Actualmente en el ámbito de herramientas de software para análisis de datos utilizadas, existen claros líderes y alternativas que podrían llenar el vacío identificado en la industria minera. A continuación se presentan las alternativas existentes que más se acercan al objetivo deseado:

- **PowerBI** [3] : Herramienta de Microsoft para análisis y visualización de datos, con capacidad para conectar diversas fuentes de datos y crear informes interactivos.
- **Seeq** [4]: Plataforma de análisis avanzado para datos industriales, especializada en análisis y visualización en tiempo real para entornos industriales.
- **Streamlit** [5]: Framework de Python [6] para desarrollar rápidamente aplicaciones web interactivas, destacando por su versatilidad y simplicidad.
- **Dash/Plotly** [1]: Combinación de Plotly [7], una biblioteca de visualización de datos en Python, y Dash, un framework para aplicaciones web, ambos enfocados en interactividad y gráficos avanzados.

1.1.1. Tabla comparativa de soluciones

En el siguiente segmento, presentamos una tabla comparativa que resume las diferencias y similitudes clave entre las diversas herramientas y tecnologías abordadas en este trabajo. Esta tabla ofrece una visión panorámica que facilita la comprensión de las

características distintivas de cada herramienta, su aplicabilidad en distintos contextos. Cada fila detalla atributos específicos, mientras que las columnas representan las herramientas analizadas.

Característica	Dash/plotly	PowerBI	Seeq	Streamlit
Interoperabilidad	✓	✗	✓	✓
Personalización	✓	✓	✓	✓
Anotaciones integradas	✓	✗	✓	✗
Costos de operación	✓	✗	✓	✓
Solución offline	✓	✗	✗	✓
Portabilidad	✓	✗	✗	✓
Código Abierto	✓	✗	✗	✓

La tabla comparativa claramente indica que la solución ofrecida por Dash/Plotly es la más completa. Por esta razón, fue la elegida para su implementación en este trabajo, destacándose por su amplia gama de funcionalidades y su adaptabilidad a nuestras necesidades específicas.

1.1.2. Objetivo principal

Los objetivos generales de esta propuesta son optimizar el tiempo utilizado para extraer y analizar datos dentro de la industria de procesos productivos, estandarizar el lenguaje visual de los reportes generados y el flujo de trabajo de los usuarios finales.

1.1.3. Objetivos específicos

Para satisfacer los objetivos generales se han propuesto los siguientes objetivos específicos:

- Implementar una forma de adquirir datos desde diversas fuentes.
- Implementar un motor de Visualización de Datos que permita la integración de

anotaciones.

- Desarrollar una Interfaz Gráfica de Usuario en lo posible intuitiva.
- Realizar pruebas funcionales y de estrés para evaluar las capacidades de importación y visualización de datos de la aplicación.

1.2. Alcances, contribuciones y estructura del documento

El trabajo se centra en el desarrollo de una herramienta de software con el único propósito de realizar la visualización y análisis de datos para aplicaciones de control avanzado.

El enfoque principal del proyecto es práctico, con el propósito de poner a prueba los conceptos teóricos y verificar la factibilidad de lograr un análisis más eficiente de datos, en lugar de centrarse exclusivamente en investigaciones teóricas o en la innovación. Sin embargo, se deja espacio para futuros desarrollos y mejoras de la herramienta.

Se espera que el proyecto sea una contribución significativa a la eficiencia y estandarización de las tareas de visualización y análisis de datos en Kairos Mining / Honeywell Chile, facilitando la toma de decisiones basadas en datos y mejorando la productividad.

Esta memoria se estructura de la siguiente manera: se presenta en los siguientes capítulos: el Capítulo 2 describe el diseño y análisis de la solución propuesta; el Capítulo 3 detalla la implementación de la herramienta; el Capítulo 4 analiza los resultados obtenidos y, finalmente, el Capítulo 5 concluye el trabajo y sugiere direcciones futuras para la investigación.

2. Diseño y análisis de la solución

2.1. Características de la herramienta

En base a las necesidades del contexto expuesto, en conjunto con el product owner[1]se definieron los requerimientos de la aplicación en dos listas: una que describe los requerimientos de software y otra que describe los requerimientos del usuario.

2.1.1. Requerimientos del usuario

A continuación se describen las acciones que el usuario debiese poder hacer con la (o en la) aplicación.

- Cargar datos de su computador a la herramienta.
- Buscar e importar datos desde un servidor.
- Visualizar y editar los datos cargados a la herramienta mediante un editor de tablas.
- Exportar datos importados y/o editados.
- Graficar los datos cargados, mediante dispersión, lineas, diagramas de cajas e histogramas.
- Reordenar y cambiar el tamaño de los gráficos creadas.
- Agregar anotaciones de texto y dibujos a los gráficos creados.
- Cambiar los atributos de los gráficos (colores, títulos, nombres de ejes, tamaño de letras, etc.)
- Agregar marcas de agua a los gráficos.
- Añadir cuadros de texto e imágenes al área de trabajo.

[1]Encargado de la visión del proyecto en representación de Honeywell Chile

2.1.2. Requerimientos de software

A continuación se detallan las características técnicas que debe cumplir la aplicación.

- Debe funcionar en un ambiente de 64bit mayor o igual a Windows 8 [8] o Windows Server 2012 R2 [9]
- La herramienta debe ser portátil (no requerir instalación o configuración por parte del usuario).
- Debe tener un tamaño reducido (inferior a 200MB).
- La herramienta debe poder funcionar off-line (sin conexión a internet o a algún servidor de datos).
- Debe conectarse a bases de datos PHD [2] y MongoDB. [11]
- Debe importar datos desde archivos .xlsx [12] y .csv [13]

2.2. Arquitectura de la aplicación

A nivel general la arquitectura propuesta es relativamente simple, ya que se abstrae del clásico modelo de vista controlador, y opta por una estructura monolítica en la que todos los componentes de la aplicación están dentro del mismo proceso y no dependen de servicios externos para funcionar. Debido a esto no se distinguen lenguajes de front-end y back-end.

Todo esto con el objetivo de simplificar la estructura y acortar los tiempos de desarrollo.

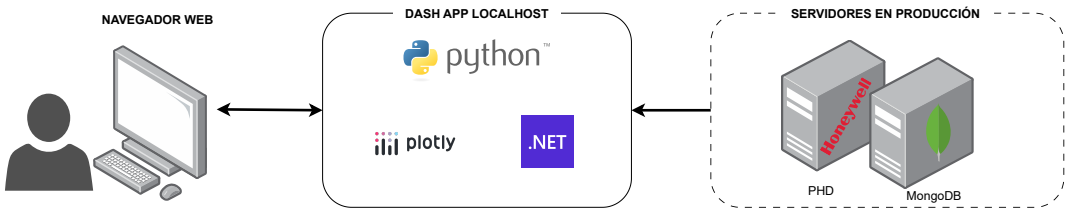


Figura 2.1: Estructura general de la herramienta

[2] Process History Database.[10]

Como se ve en la figura 2.1 la herramienta está contenida en un solo proceso, en el cual se leen datos de los servidores en producción, que recopilan datos históricos de procesos productivos.

2.3. Diseño de las interfaces

El diseño de las interfaces de esta aplicación fue un trabajo altamente iterativo, inspirado por el diseño de otras alternativas del mercado, pero fuertemente dictado por las necesidades del usuario, además de la naturaleza de los componentes utilizados, ya que al utilizar un framework de alta abstracción, muchos de los componentes no pueden ser modificados.

2.4. Tecnologías utilizadas

Para desarrollar las funcionalidades propuestas en este trabajo, se emplearon múltiples desarrollos de otros autores. A continuación, se presentan las decisiones tomadas en este proceso.

2.4.1. Back-end

Para abordar este proyecto, se decidió desarrollar una herramienta monolítica y autocontenida utilizando Python [6], debido a su simplicidad y popularidad dentro de la industria. La elección de Python no solo se basa en su facilidad de uso, sino también en su capacidad para escribir una aplicación que, aunque de momento no es necesario, tiene actualmente la capacidad de ser multiplataforma. Esta decisión proporciona flexibilidad y potencial escalabilidad a la herramienta, asegurando su adaptabilidad a futuros requerimientos y entornos tecnológicos.

2.4.2. Gráficos e interfaz gráfica

Dash [1] es la piedra angular que posibilita que esta herramienta sea escrita completamente en Python[6]. Se eligió debido a su capacidad para abstraer HTML [3], React [15] y Flask[16] en componentes y clases de Python. Todo esto se encuentra empaquetado en un framework poderoso que convierte a Python en un lenguaje full-stack. Desarrollado por el equipo que creó Plotly[7], Dash[1] ofrece soporte nativo para todos los componentes de análisis de datos de Plotly, lo que lo convierte en la solución ideal para este trabajo.

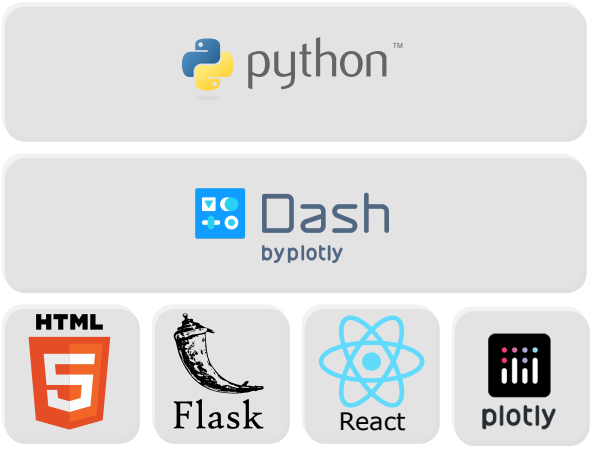


Figura 2.2: Stack de abstracción de tecnologías

Adicionalmente para complementar Dash[1] se requieren librerías externas.

`dash-bootstrap-components` (dbc) [17] es una librería que nos permite utilizar la mayoría de los componentes de bootsrap (un framework de componentes de css), dentro de nuestro proyecto, solo usando python.

`dash-mantine-components` (dmc) [18] es una librería de componentes similar a dash-bootstrap-components[17] pero para los componentes de mantine (una librería de componentes para react native). [10] `dash-draggable` (dd) [19] es una librería que nos permite agregar funcionalidad "drag and drop" dentro de la herramienta. Permite crear

[3] HyperText Markup Language. [14]

lienzos, dentro de los cuales cualquier componente añadido puede ser arrastrado y transformado.

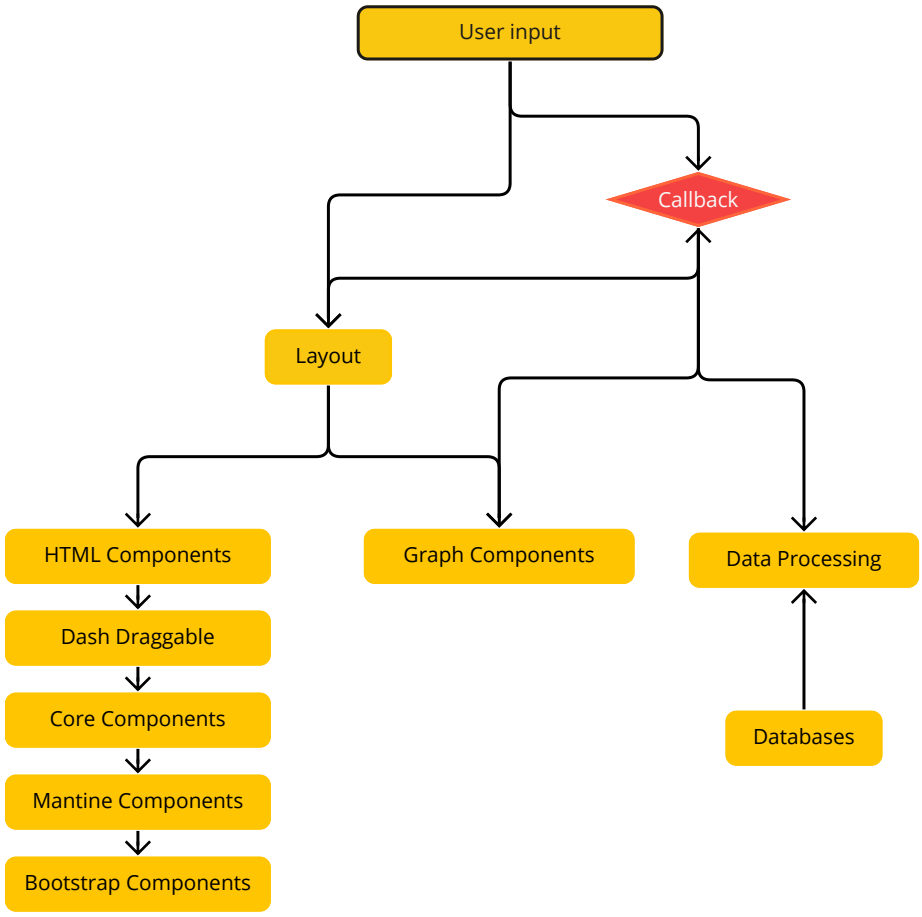


Figura 2.3: Diagrama de interacciones de los componentes

3. Implementación de la solución

3.1. Estructura del proyecto

El proyecto se encuentra dividido en carpetas con nombres que representan la funcionalidad que tienen dentro del proyecto. La jerarquía está estructurada de tal forma que el archivo principal `index.py` y el resto de los archivos Python[6] estén en la máxima altura en la estructura de archivos, con el objetivo de simplificar las referencias a archivos o carpetas.

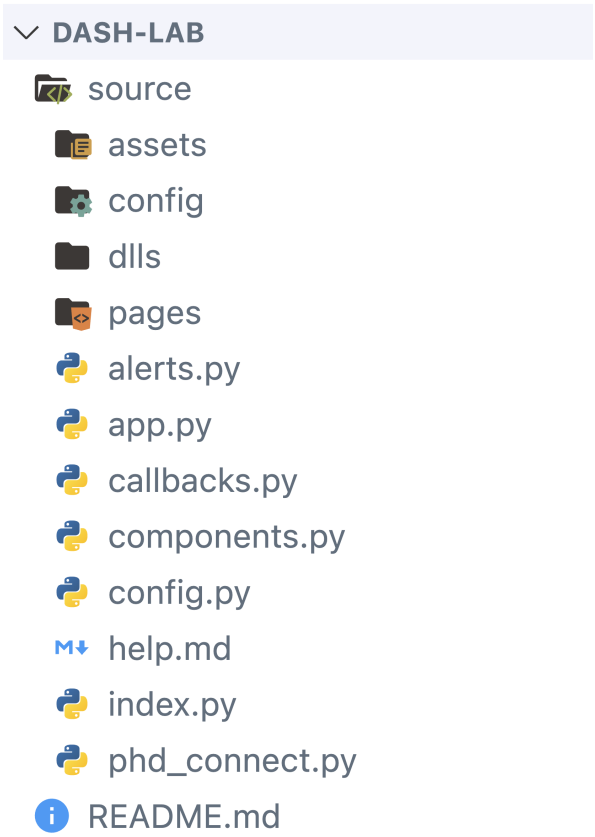


Figura 3.4: Estructura de archivos del código del proyecto

`\assets`: Contiene todos los archivos visuales que se utilizan en la aplicación.

`\config`: Contiene todas las variables de configuración en formato de archivos json. Adicionalmente contiene la carpeta de archivos de lenguajes, los cuales también son

diccionarios json (ver Figura 3.5).

Figura 3.5: Estructura de la carpeta de configuración

`\dlls`: Contiene los archivos dll[4] necesarios para conectarse al PHD[10] (ver Figura 3.6).

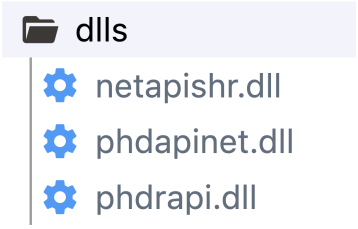


Figura 3.6: dlls usadas por la herramienta

`\pages`: Es una carpeta solicitada por la documentación de Dash[1] para incluir la funcionalidad de paginación a la aplicación. Dentro de esta se pueden incluir archivos que describan una vista completa. Estas vistas pueden ser usadas en cualquier parte de la aplicación, por si solas o dentro de otra vista (ver Figura 3.7).

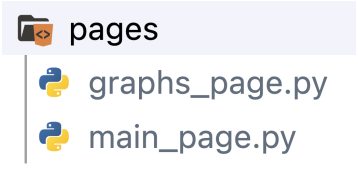


Figura 3.7: Carpeta de páginas

[4]Dynamic-link library [20]

`alerts.py`: Es el archivo donde se definen todas las notificaciones del sistema.

`app.py`: Solo se utiliza para definir la instancia de la aplicación y es importado por callbacks.py, components.py e index.py.

`callbacks.py`: Se definen todos los callbacks[21] de Dash [1] y por lo tanto es donde ocurre toda la lógica de la aplicación. Se podría llamar al archivo de callbacks como el back-end de este desarrollo, pero por la naturaleza del framework Dash, gran parte de la interfaz gráfica se define en los callbacks.

`components.py`: Es un archivo donde se definen componentes de la interfaz gráfica, para que puedan ser utilizados por los callbacks[21].

`config.py`: Es el archivo que se encarga de manejar los archivos de configuración.

`index.py`: Es el archivo donde se define la vista principal de la aplicación y donde se levanta la instancia del servidor local.

`phd_connect.py`: Es un archivo que se encarga de ser la interfaz entre la dll que lee el phd y el resto del proyecto.

3.2. Ingesta de datos

Para poder leer tags de una base de datos PHD [10], se necesitan dos cosas: phdapinet.dll (dll[20] suministrada por Honeywell) y pythonnet [22] (librería de Python que nos permite interactuar con dlls .NET). Con estos dos elementos se usó la herramienta dotpeek [23] para decompilar la dll, y así ver su funcionamiento interno. Guiándose por el trabajo expuesto en [24], fue posible crear la clase de `get_all_tags_list(serverIpAddress)` para obtener la lista de todos los tags almacenados dentro de una base de datos phd. Y la clase `Get_tags(TagsList)`, la cual retorna un dataframe [25] con los datos asociados a cada tag de cualquier lista de tags provista por el usuario.

```
def Get_All_Tags_list(PDH_server)
    tagsList = PDH.AllTags
```

```
        return tagsList


    def Get_Tags_Data(TagsList)
        TagsData = Pandas.Dataframe(PHD.TagsList)
        returnTagsData
```

La conexión a las bases de datos MongoDB [11] se realizó utilizando la librería PyMongo[26].

3.3. Implementación de interfaces mediante componentes

Como fue mencionado en la Sección 2.4.2 el uso de las librerías, dcc, dbc y dmc permite crear interfaces modulares y reutilizables mediante el uso de componentes. Un componente es una clase de Python que en su capa mas básica contiene la definición de una interfaz de HTML [14]. Los componentes pueden tener atributos de identificación, valor, estado y componentes hijos.

Todos los elementos de la interfaz gráfica de la aplicación desarrollada son un componente o son parte de un componente más grande.

Como ejemplo de la implementación de un componente, a continuación se muestra el código en Python para crear los componentes `add_graph_button` y `add_text_block` los cuales forman parte de un componente mayor.

```
add_single_graph_button = html.Div([
    dmc.Button(lang["main-page"]["left-area"]["buttons"]["add-graph"],
               id='add-single-graph-button', variant='outline'),
    ], className='flex-fill',)
```

```
add_text_block = html.Div([
    dmc.Button(lang["main-page"]["left-area"]["buttons"]["add-text-block"],
               id='add-text-block-button', variant='outline'),
    ], className='flex-fill',)
```

Los dos componentes anteriores son parte del componente mayor `left_area` como se muestra a continuación:

```
left_area = html.Div(
    id='left-area',
    children=[
        html.Div(id='left-pane-header', children=[
            html.H6(lang['main-page']['left-area']['labels']
                    ['title'], className='text-left'),
            dmc.Group([
                add_single_graph_button,
                add_text_block,
            ],spacing=5, className='flex-fill'),
        ]),
    ],)
```

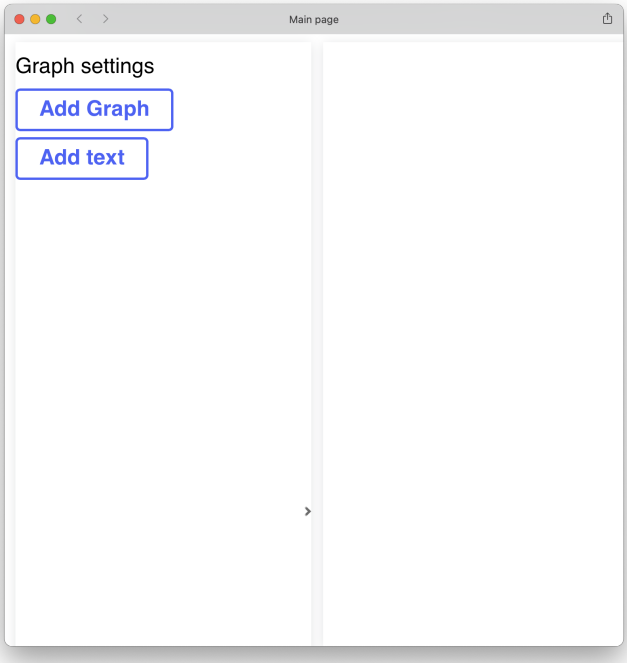


Figura 3.8: Ejemplo de componente "left area"

3.4. Implementación de interactividad mediante callbacks

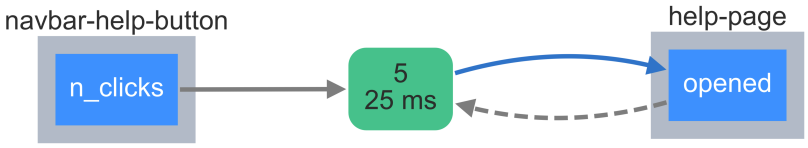


Figura 3.9: Diagrama del callback para abrir y cerrar la página de ayuda

Los callbacks [21], que son funciones esenciales, se activan cuando el usuario realiza una acción o cuando un elemento del sistema cambia su estado. Para definir un callback correctamente, es necesario especificar tres aspectos: qué componente estará 'observando', qué funcionalidad ejecutará, y cuál componente recibirá el resultado de la función ejecutada.

Como se ve en la Figura 3.10, la aplicación desarrollada requiere una gran cantidad de interacciones.

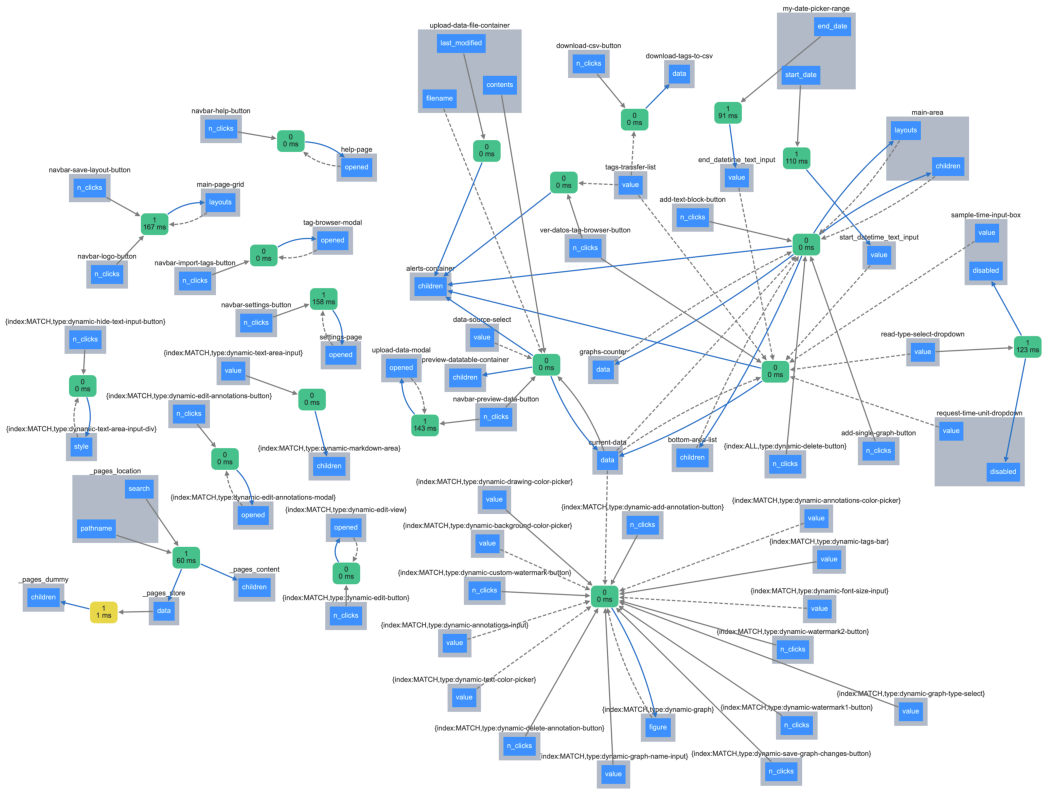


Figura 3.10: Captura del diagrama de interacciones de callbacks, obtenida del debugger de Dash[1].

A continuación, se muestra un trozo de código que describe el callback[21] para abrir y cerrar la página de ayuda. Como se puede apreciar en la Figura 3.9, el callback es representado por un cuadrado verde donde se puede ver cuántas veces se ha ejecutado y cuánto demoró la ultima interacción. También en la Figura se puede ver cómo el callback altera el estado de la página de ayuda, y al mismo tiempo lo lee.

```
# Open and close help page
    @app.callback(
        Output("help-page", "opened"),
        [Input("navbar-help-button", "n_clicks"),],
        [State("help-page", "opened")],
        prevent_initial_call=True
    )
    def edit_graphs(help_button, opened):
        if help_button:
            return not opened
        else:
```

```
            raise PreventUpdate
```

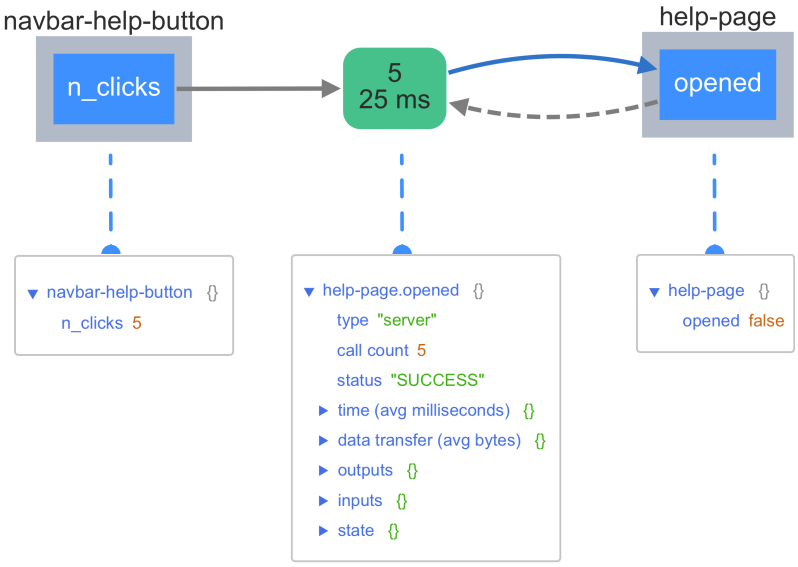
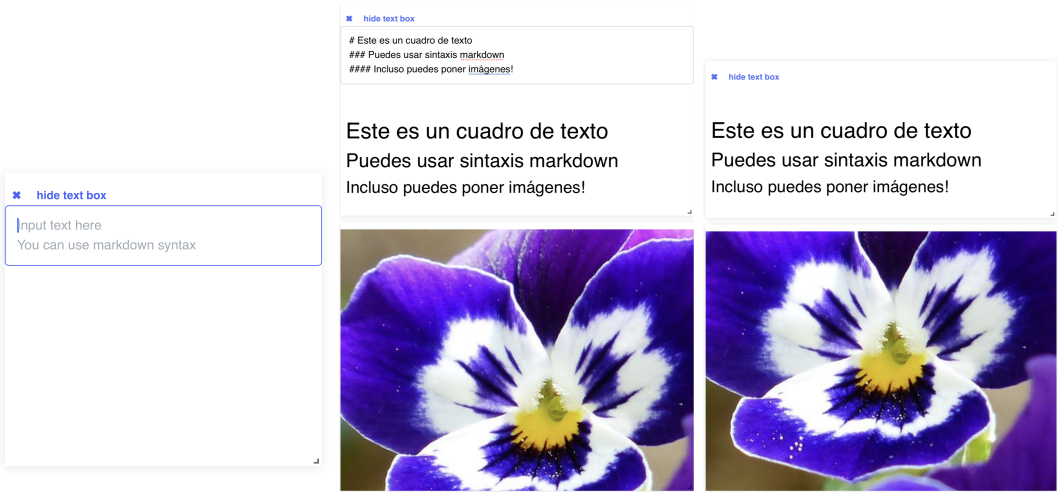


Figura 3.11: Diagrama del callback para abrir la página de ayuda con sus estados.

3.5. Implementación de cuadros de texto



(a) Cuadro vacío. **(b)** Modo edición. **(c)** Modo vista.

Figura 3.12: Ejemplo del sistema de cuadros de texto.

Los cuadros de texto permiten al usuario agregar cualquier texto en formato Markdown [27] y visualizarlo de manera rápida; además, el contenedor facilita la adición de imá-

genes, enlaces y listas. Estos cuadros de texto se crean insertando un componente de entrada de texto y conectándolo a otro componente que interpreta Markdown mediante un callback[21].

A continuación se presenta un ejemplo funcional de cómo se declara una aplicación que solo contiene un cuadro de texto.

```
import dash
from dash import html, dcc
from dash.dependencies import Input, Output

# Create a Dash application
app = dash.Dash(__name__)

# Define the layout of the app
app.layout = html.Div([
    dcc.Textarea(id='text-input', placeholder='Enter text here...'),
    html.Br(),
    dcc.Markdown(id='markdown-output')
])

# Callback to update Markdown content based on text input
@app.callback(
    Output('markdown-output', 'children'),
    [Input('text-input', 'value')]
)
def update_output(value):
    if not value:
        return 'Enter text in the input above to see it here.'
    return value

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

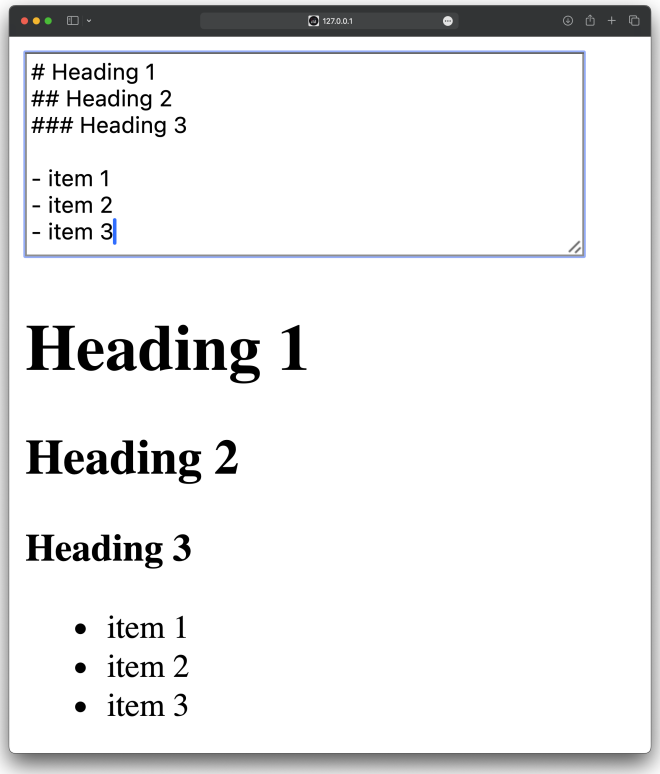


Figura 3.13: Ejemplo básico de cuadro de texto.

3.6. Implementación de gráficos

Como se menciona en la Sección 2.4.2, Plotly [7] es el encargado de facilitar la creación de gráficos dentro de la aplicación.

En la Figura 3.14, se pueden apreciar algunas de las funcionalidades adicionales que Plotly [7] ofrece: todas las etiquetas pueden ser editadas haciendo doble clic sobre ellas, incluyendo el título, el eje vertical, el eje horizontal y los nombres en la leyenda. Asimismo, incorpora un cursor que indica el valor de las variables en esa posición específica. Además, cuenta con una barra de opciones que incluye controles de zoom, paneo, autoescala, selección, dibujo y exportación de la gráfica en formato PNG[5].

[5]Portable Network Graphics

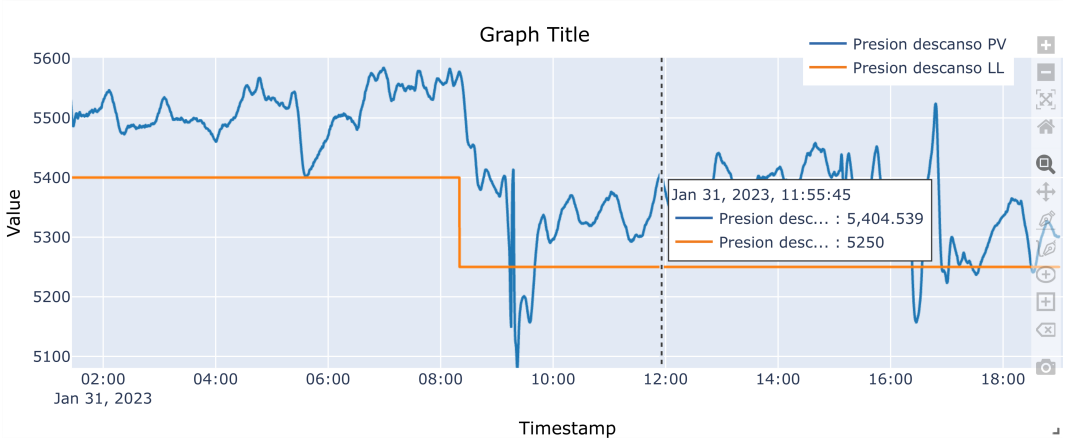


Figura 3.14: Ejemplo de gráfica con barra de opciones, cursor y leyenda.

A continuación se presenta un ejemplo funcional en el que se detalla cómo crear un gráfico básico en Plotly. Primero se debe definir un "graph object" al cual se le debe modificar el atributo "Figure": este contiene el tipo de gráfico y los datos. Luego, el "graph object" es pasado al atributo figure de un objeto `dcc.Graph()`

```
import dash
from dash import dcc
import Plotly.graph_objects as go
import numpy as np

app = dash.Dash(__name__)
server = app.server

N = 1000
t = np.linspace(0, 10, 100)
y = np.sin(t)

fig = go.Figure(data=go.Scatter(x=t, y=y, mode='markers'))



app.layout = html.Div([
    graph = dcc.Graph(
        id='example-graph',
        figure=fig
    )
])
```

```
if __name__ == '__main__':
    app.run_server(debug=True, port='8080')
```

3.7. Implementación de layout configurable.

Figura 3.15: Ejemplo de ventanas transformables

Como se menciona en la Sección 2.4.2, el uso de la librería `dash_draggable` (dd) [19] nos permite implementar ventanas móviles y transformables. Esto permite que el usuario interactúe de manera más natural con la herramienta. Si desea cambiar el tamaño de algún elemento después de haberlo creado, solo debe arrastrar desde la esquina o el borde, donde se encuentran los indicadores de agarre. Por otro lado, si desea mover la ventana a otra posición, solo debe tomarla desde la parte superior.

Como se observa en la Figura 3.15, es posible generar ventanas móviles dentro de otras ventanas. En este ejemplo, se muestra cómo las ventanas que contienen gráficos y cuadros de texto están encapsuladas dentro de la ventana más grande. Además, en la Figura 3.15, se evidencia que todas las ventanas poseen un indicador en la esquina inferior derecha, mientras que solo las tres más grandes cuentan con indicadores en los

dos lados adyacentes. Estos indicadores alertan que las ventanas pueden ser arrastradas desde allí para cambiar su tamaño.

A continuación se presenta un mínimo ejemplo funcional en Python de las ventanas de la librería Dash draggable [19].

```
import dash
import dash_html_components as html
import dash_draggable

app = dash.Dash(__name__)

server = app.server

app.layout = html.Div([
    dash_draggable.ResponsiveGridLayout(
        children=[
          html.H1("Window 1"),
          html.H1("Window 2"),
          html.H1("Window 3"),
        ]
        )
])

if __name__ == '__main__':
    app.run_server(debug=True, port='8080')
```

3.8. Vistas e interfaces de usuario

3.8.1. Página principal

La página principal es donde el usuario pasará la mayor parte del tiempo, ya que es donde se encuentran contenidos los gráficos y cuadros de texto.

Como se visualiza en la Figura 3.16, al iniciar la página principal, ésta comienza como un lienzo vacío, y está compuesta por cuatro componentes principales: una barra de navegación estática, y tres ventanas móviles, las cuales pueden ser transformadas en tamaño y reordenadas de la manera en que el usuario encuentre adecuada. El usuario también tiene la capacidad de guardar el estado actual de las ventanas mediante el

botón cuadrado que se encuentra en la derecha, en la barra de navegación. Este botón permite que la disposición actual sea cargada cada vez que la página se refresque, incluso si la aplicación se cierra y su proceso es eliminado.

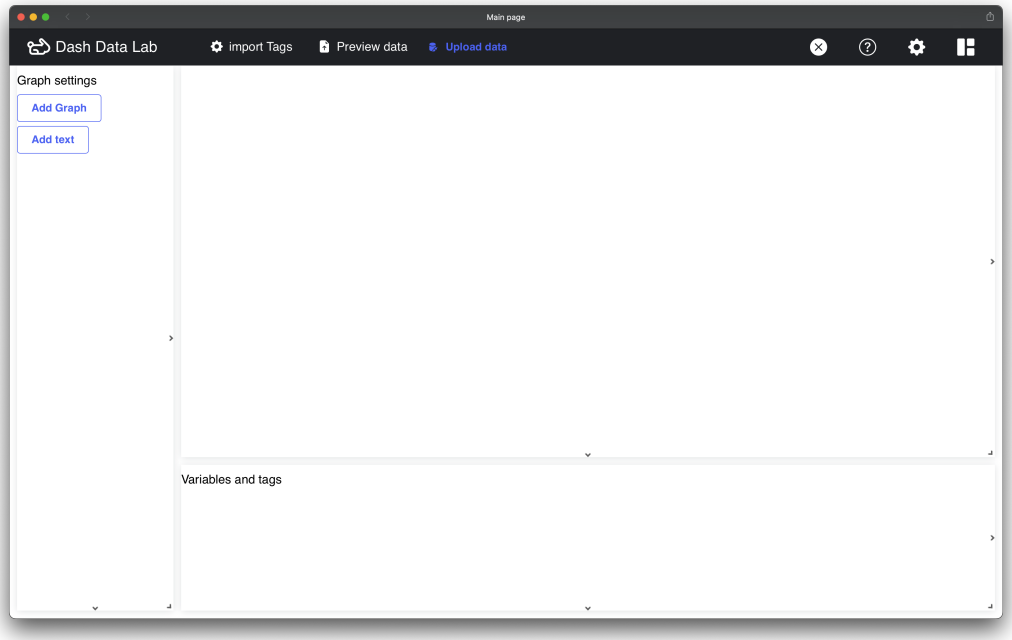


Figura 3.16: Interfaz de la página principal vacía.

La barra de navegación incluye botones para acceder a la mayoría de las otras interfaces de usuario de la aplicación, tales como: el explorador de etiquetas, el visualizador de datos, el buscador de archivos, la página de ayuda y la página de configuración. Además, la barra de navegación cuenta con funcionalidades adicionales, como la capacidad de guardar la distribución de ventanas actual mediante el botón ubicado más a la derecha, así como la opción de volver a la disposición inicial a través del botón situado más a la izquierda. También, se ofrece la posibilidad de cargar y procesar automáticamente un archivo al arrastrarlo desde cualquier parte del sistema operativo hasta el botón "upload data".

3.8.2. Área de trabajo

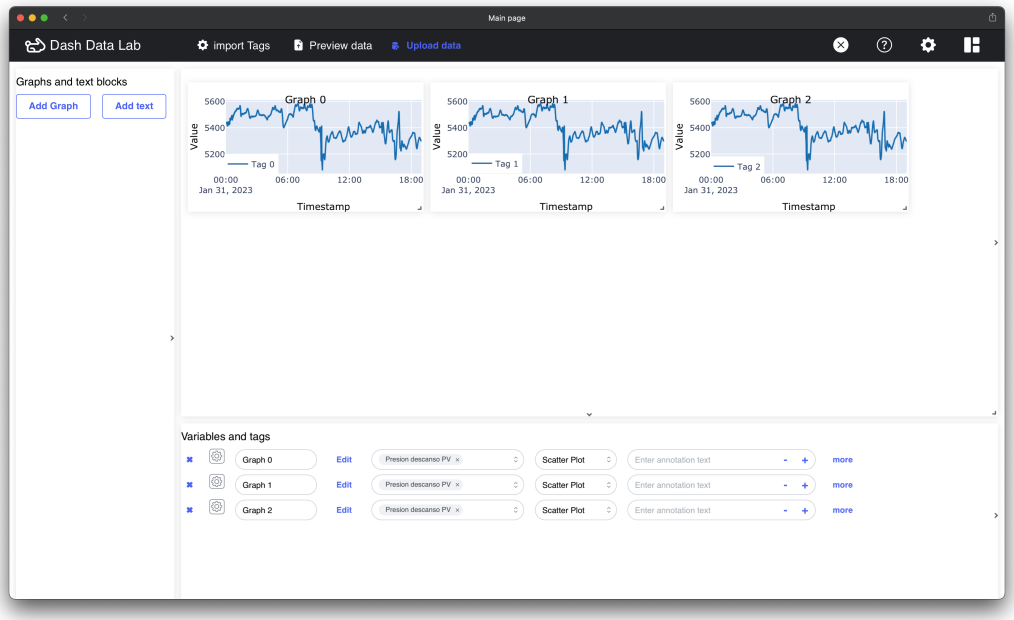


Figura 3.17: Ejemplo de 3 gráficos iguales.

El área más grande se denomina área principal, porque es el contenedor de todos los gráficos y cuadros de texto. Todo lo que se agregue a esta área principal también será movible y transformable en tamaño.

La ventana izquierda contiene los botones para agregar ventanas de gráficos o bloques de texto al área de trabajo.

El área inferior contiene las opciones individuales para cada gráfica que se presente en al área de trabajo. Estos elementos no son transformables ni movibles, pero si pueden ser eliminados; ya que como se ve en la Figura 3.17, al presionar la **X** que se encuentra al comienzo de cualquier línea, esta línea de opciones es borrada junto con su correspondiente gráfico. Junto a estas opciones también se encuentra: una barra para cambiar el nombre de la gráfica, un botón denominado "edit" que accede a las opciones de la gráfica, un menú desplegable para seleccionar de qué forma se van a graficar los datos; y finalmente una barra para agregar anotaciones de texto, junto con un botón para

acceder a un menú para cambiar el color de las anotaciones y los dibujos de los gráficos.

En la Figura 3.18 se ve cómo se pueden graficar múltiples variables en una sola ventana, además de cómo se despliega el menú para seleccionar el tipo de gráfico.

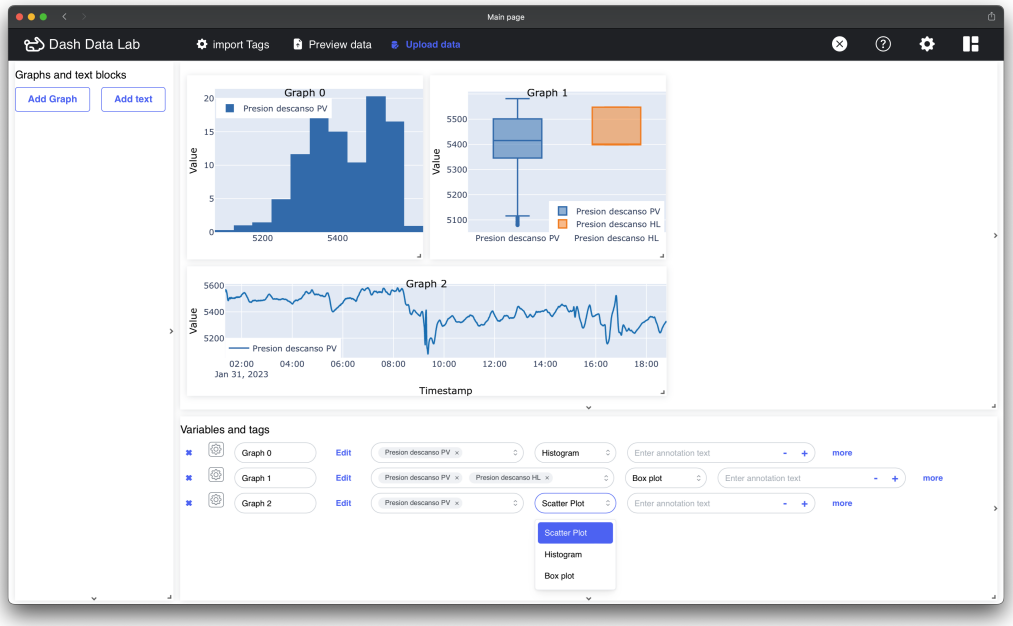


Figura 3.18: Ejemplo de 3 gráficos de diferente tipo.

3.8.3. Explorador de variables (Tag Browser)

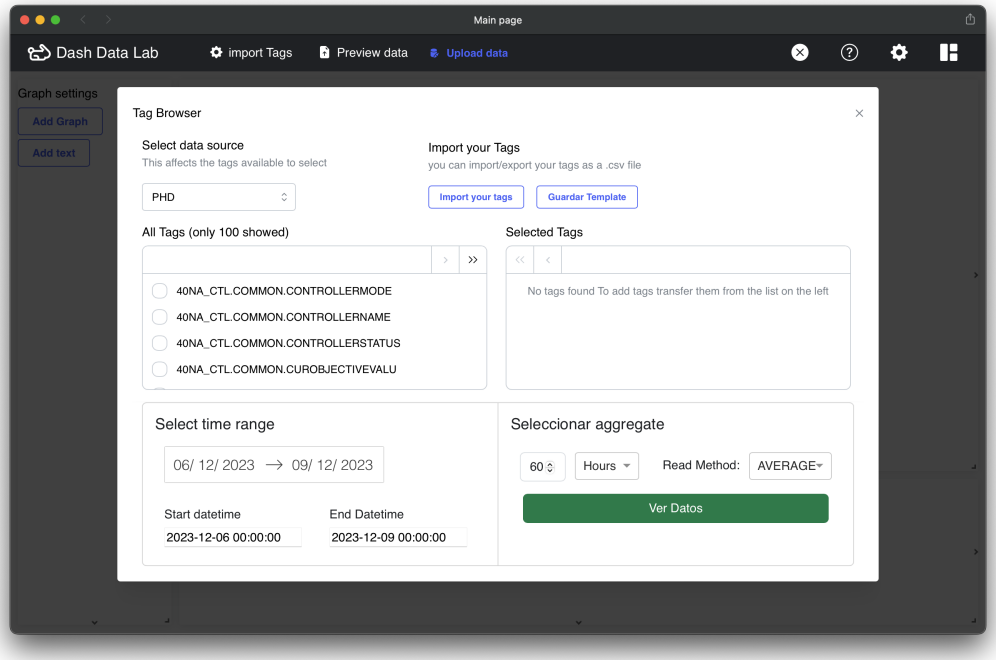


Figura 3.19: Interfaz del explorador de variables.

En esta interfaz se encuentran todas las opciones necesarias para poder leer variables desde una fuente externa. Como se puede apreciar en la Figura 3.19, al abrir la interfaz por primera vez, esta presenta dos columnas: una que contiene la lista de todas las variables del servidor y otra que está vacía. La lista de la derecha son las variables que leeremos. Las variables a consultar deben ser obtenidas a partir del listado ubicado a la izquierda (listado de variables disponibles) para posteriormente ser trasladadas hacia el listado de la derecha (variables seleccionadas). Todo esto se realiza mediante los botones que se encuentran en la parte superior de ambas listas, los cuales sirven para importar solo lo seleccionado, o todo lo que encuentra dentro de la lista.

Esta interfaz nos da la opción de seleccionar el origen de las variables, además de la posibilidad de exportar las variables seleccionadas a un archivo .csv y de importar algún archivo de variables seleccionadas previamente creado. Cabe destacar que estos

archivos solamente contienen los nombres de las variables y no los datos que estas podrían contener, por lo que al importar un archivo de variables igualmente se deben leer los datos de la fuente seleccionada. Para importar los datos se debe elegir un periodo de tiempo en el que se desee obtener datos; y para esto hay un selector de fecha con calendario, además de una vista previa del tramo con horas, minutos y segundos en la parte inferior. Adicionalmente, para leer los datos es necesario seleccionar el intervalo de muestreo y el tipo de interpolación a utilizar:

- **Average**:Calcula el promedio de los datos dentro del intervalo de muestreo seleccionado. Esta opción es útil para suavizar fluctuaciones rápidas y obtener una visión general del comportamiento de los datos en un período específico.
- **Raw**: Muestra los datos tal como se registraron, sin ningún procesamiento o agregación. Esta opción es ideal para análisis detallados que requieren precisión y para situaciones donde cada punto de datos es crucial.
- **Snapshot**: Proporciona una instantánea de los datos en un momento específico dentro del intervalo de muestreo. Es útil para examinar el estado exacto de los datos en un punto en el tiempo, lo que permite un análisis puntual y detallado.

3.8.4. Carga de datos (Upload data)

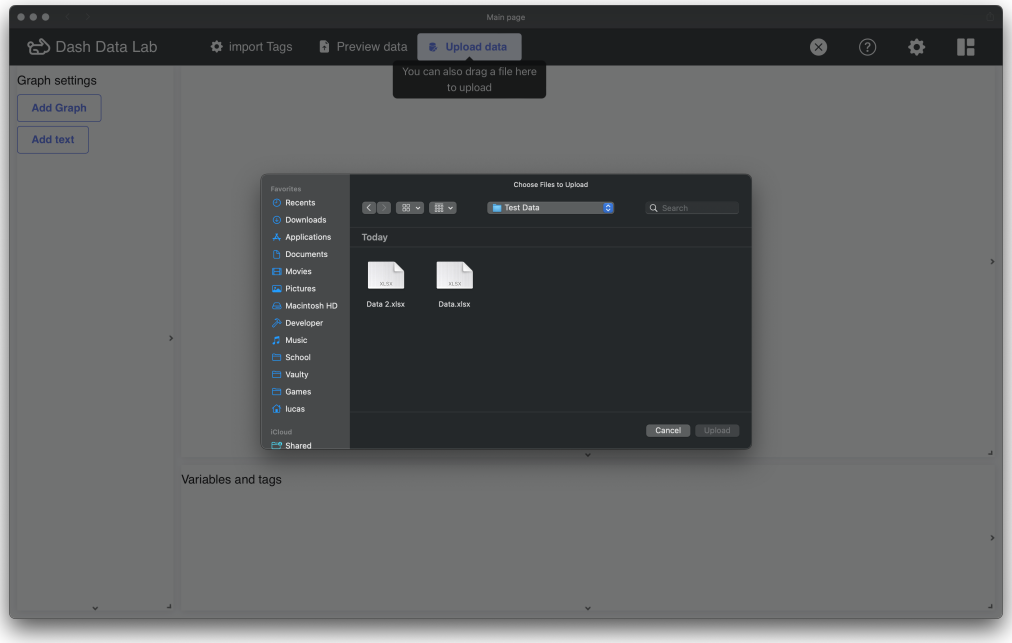


Figura 3.20: Interfaz para cargar datos desde archivo.

Esta interfaz permite al usuario cargar archivos de datos locales en formato .csv o .xlsx; Para cargar los datos, el usuario debe presionar el botón denominado "upload data" y usar el explorador de datos emergente, o simplemente arrastrar el archivo y soltarlo encima del botón de cargar datos. La herramienta leerá el archivo automáticamente.

La interfaz para cargar archivos de datos utiliza la interfaz nativa del sistema operativo en el que se ejecuta la aplicación, como se ve en el ejemplo de la Figura 3.20, donde se usa el explorador de archivos nativo de MacOS Ventura [28].

3.8.5. Visualizador y editor de datos (Data preview)

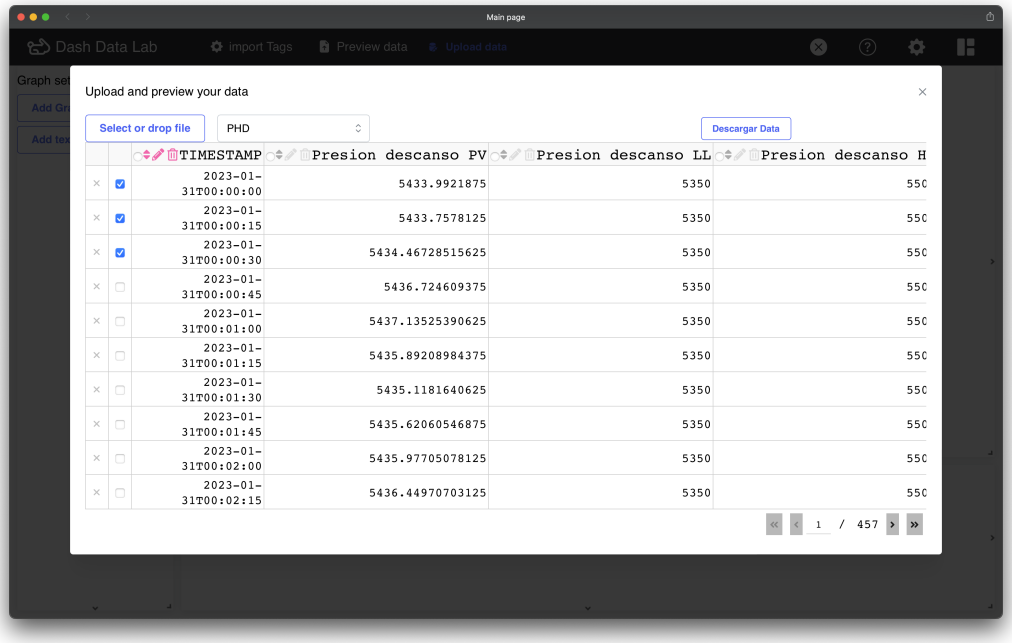


Figura 3.21: Interfaz para visualizar y editar datos.

Esta interfaz corresponde un visor de tablas de datos. La tabla visualizada puede ser editada directamente desde esta interfaz. Como se observa en la Figura 3.22, se puede cambiar fácilmente el nombre de las variables, además de editar datos puntuales o eliminar columnas y filas completas.

Como se aprecia en la Figura 3.25b se tiene nuevamente la capacidad de subir datos para ayudar al usuario, en el caso de que no lo haya hecho anteriormente. Además, se cuenta con la capacidad de exportar datos editados o importados desde alguna fuente externa.

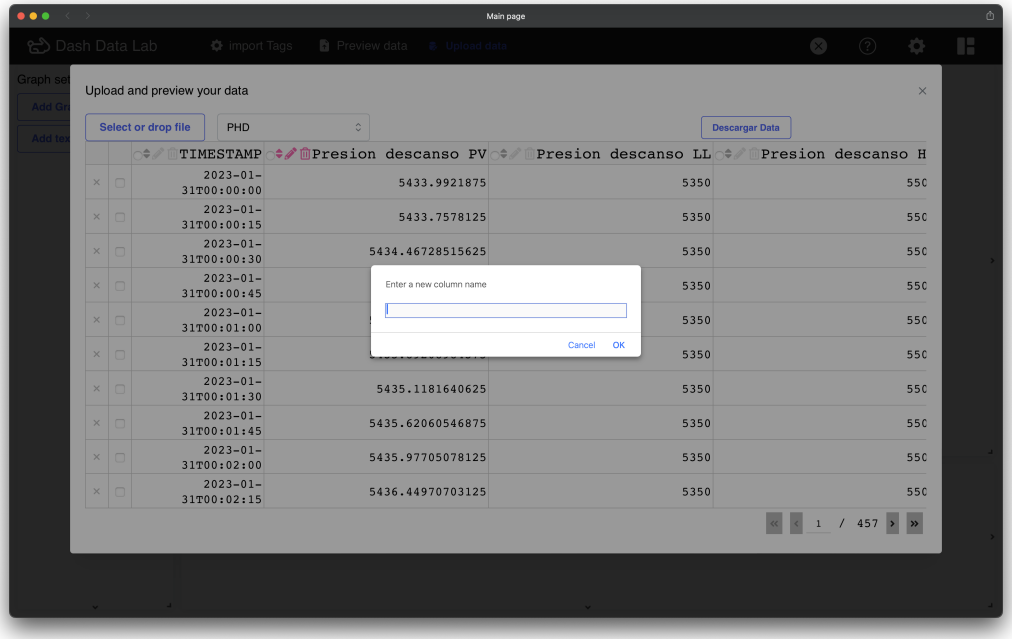


Figura 3.22: Interfaz para editar el nombre de la columna.

3.8.6. Interfaz de marcas de agua y opciones de dibujo

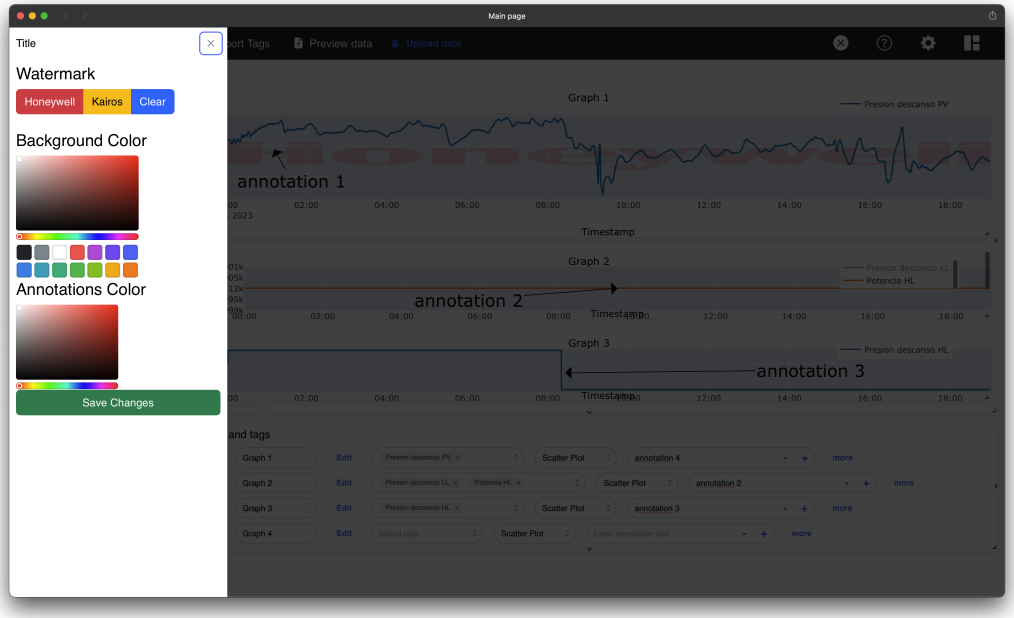


Figura 3.23: Interfaz para añadir marcas de agua y cambiar colores de los dibujos.

En esta interfaz se pueden para agregar marcas de agua al fondo de la gráfica para evitar copias no autorizadas de los análisis realizados. Además, se encuentran las opciones para editar los colores del fondo de la gráfica y de los textos.

3.8.7. Interfaz de opciones de anotaciones

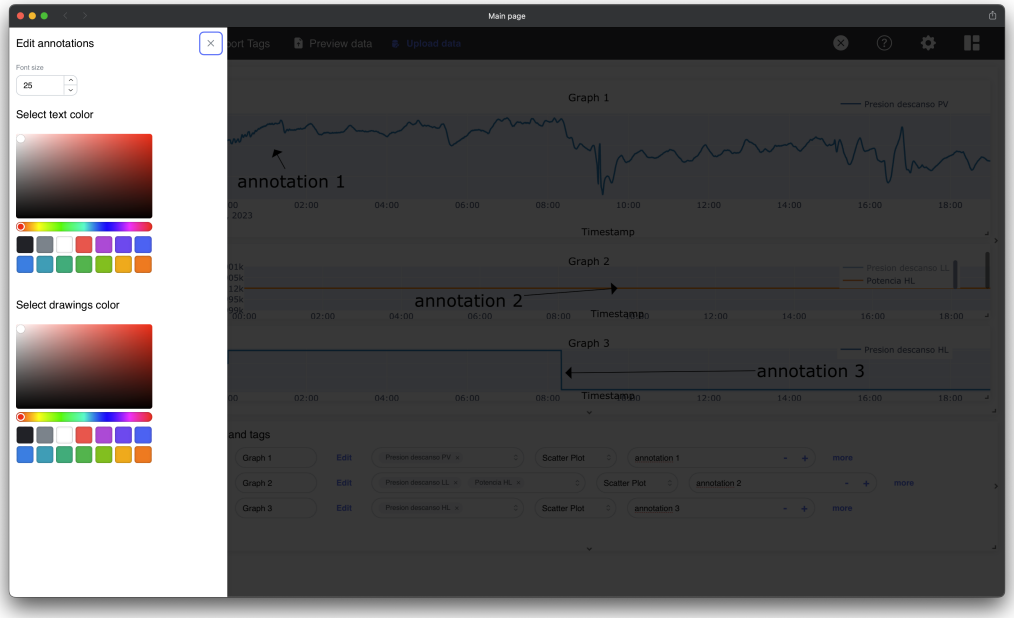
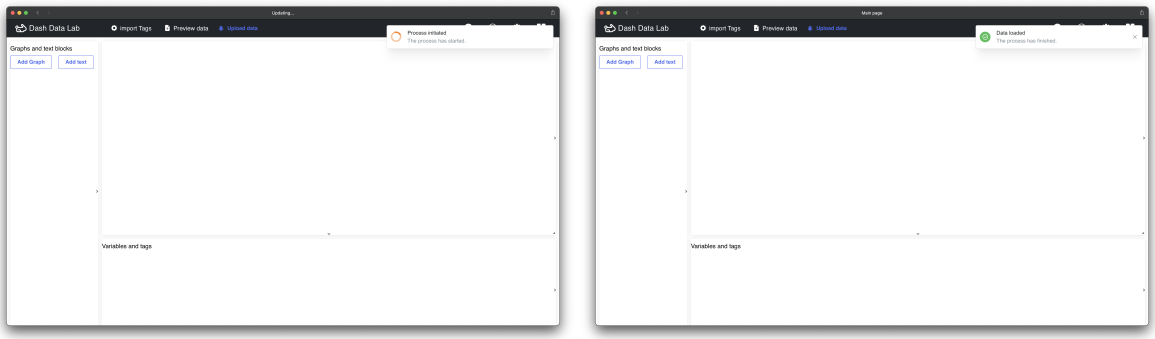


Figura 3.24: Interfaz para cambiar las opciones de las anotaciones.

En esta interfaz se encuentran las opciones de color y tamaño de letra para las anotaciones de texto, así como la posibilidad de cambiar el color de los dibujos o figuras manuales añadidos al gráfico.

3.9. Notificaciones y alertas



(a) Estado inicial. **(b)** Estado final.

Figura 3.25: Ejemplo del sistema de notificaciones con una notificación de 2 estados.

La herramienta cuenta con un sistema de notificaciones que da información importante al usuario, estas notificaciones pueden ser alertas de errores, informativas o estado de carga de algún elemento.

Para implementar estas notificaciones, se utilizó el sistema de notificaciones de dash-mantine-components [18], el cual contiene un elemento llamado "notifications provider" que debe encapsular todo el layout de la aplicación, de la siguiente manera:

```
app.layout = dmc.NotificationsProvider(
    html.Div(
        [
            html.Div(id="notifications-container"),
            dmc.Button("Show Notification", id="notify"),
        ]
    )
)
```

A continuación se muestran más ejemplos de las notificaciones de la herramienta.

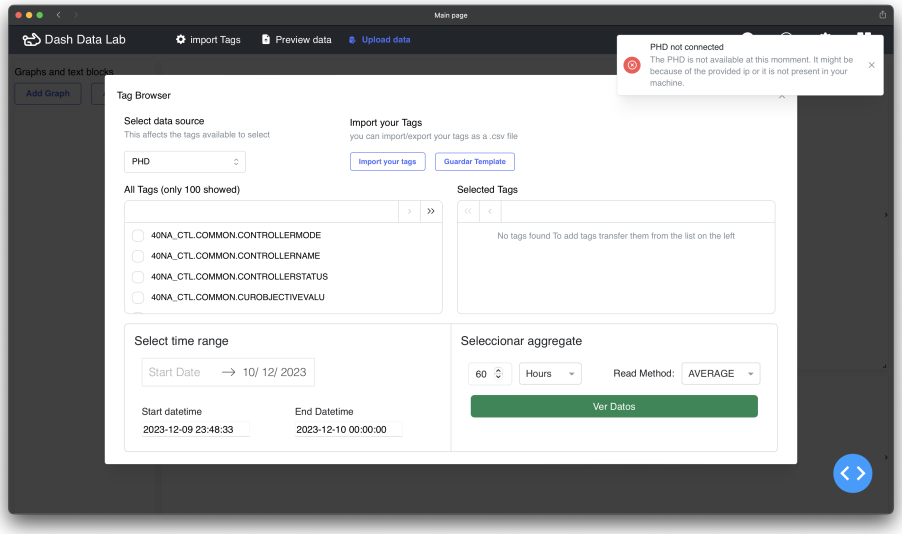


Figura 3.26: Notificación de alerta "PHD no disponible".

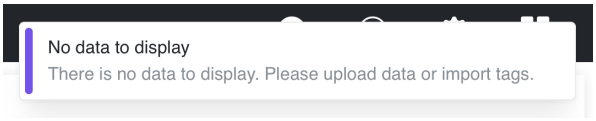


Figura 3.27: Notificación informando que no hay datos para graficar.

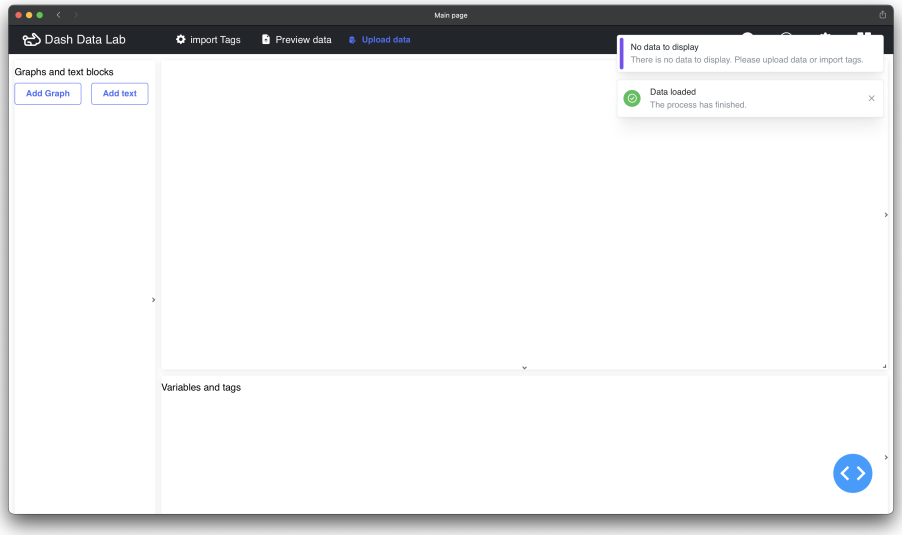


Figura 3.28: Ejemplo de cómo el sistema maneja múltiples notificaciones.

3.9.1. Página de ayuda

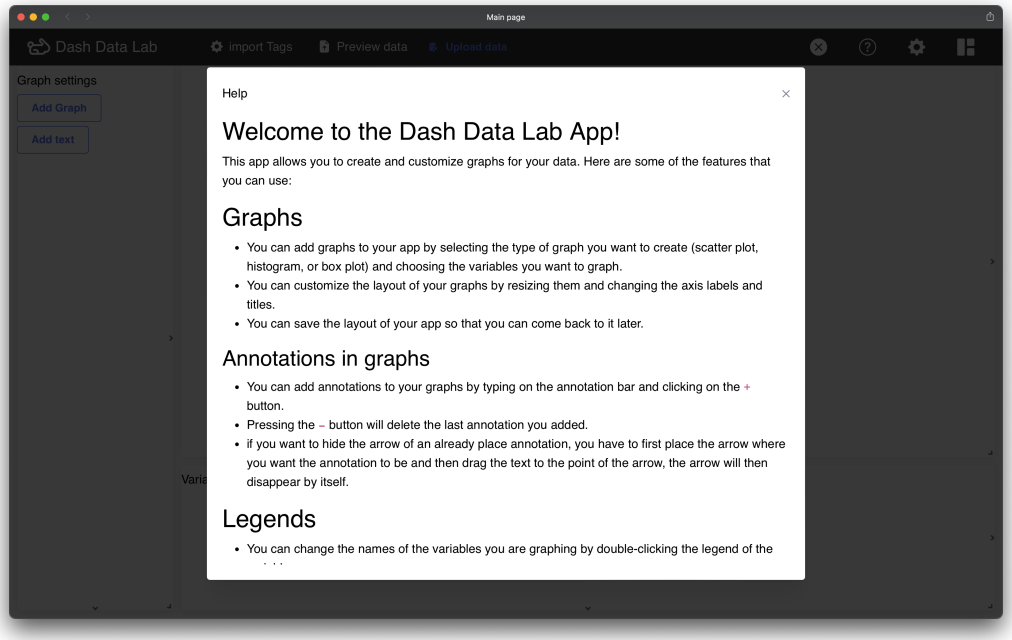


Figura 3.29: Página de ayuda de la herramienta.

La página de ayuda puede ser accedida desde la barra de navegación, presente en la página principal y que consiste en una ventana emergente, la cual contiene una página de consejos e información útil para el usuario.

3.9.2. Página de configuraciones

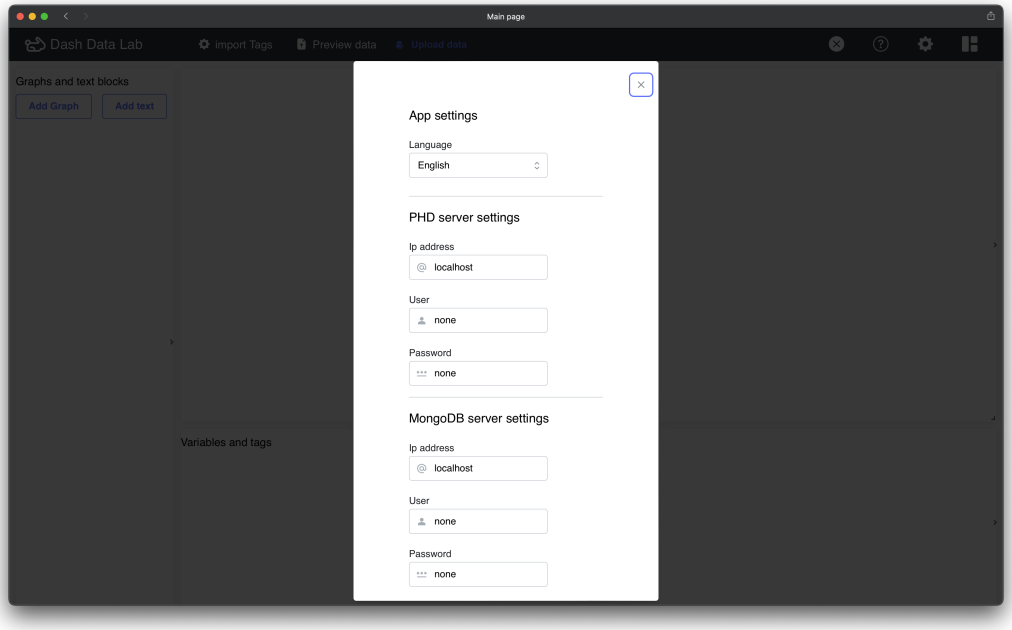


Figura 3.30: Página de configuraciones de la herramienta.

La página de configuraciones, al igual que la página de ayuda (ver Figura 3.29) es una ventana emergente flotante, la cual contiene configuraciones generales para toda la aplicación, como el idioma y los parámetros de configuración para conectarse a las diferentes fuentes de datos.

3.10. Implementación de anotaciones y dibujos

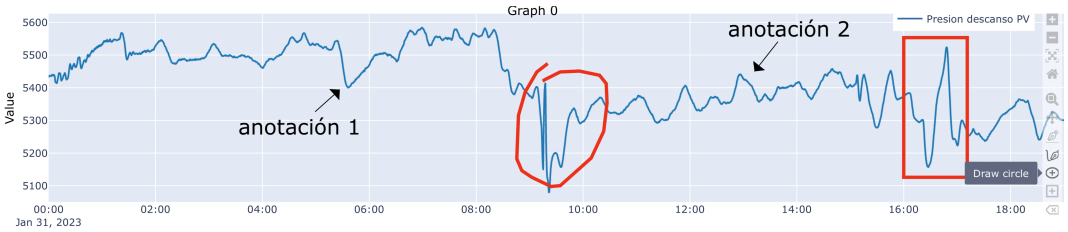


Figura 3.31: Ejemplo de anotaciones en un gráfico.

Las anotaciones son una parte importante de la herramienta desarrollada, ya que permiten añadir indicadores visuales fácilmente a los gráficos generados.

Para habilitar las anotaciones dinámicas (añadir, eliminar y transformar en cualquier momento) es necesario habilitar las siguientes configuraciones en la gráfica:

```
    config={
    'editable': True,
    'edits': {
        'annotationPosition': True
    }
```

A continuación se presenta un ejemplo funcional de un gráfico y un cuadro de texto para poder agregar anotaciones.

```
import plotly.express as px
from dash import Dash, html, dcc, Input, Output, State

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
y = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

# Initialize app
app = Dash(__name__)
initial_figure = px.scatter(x=x, y=y)

# App layout
app.layout = html.Div([
    dcc.Input(id='input_annotations', type='text',
              placeholder='Enter annotation'),
    html.Button('Add annotation', id='btn_submit'),
    dcc.Graph(id='graph1', figure=initial_figure, config={'editable': True})
])

# Callback to update graph with annotations

@app.callback(
    Output('graph1', 'figure'),
    [Input('btn_submit', 'n_clicks')],
    [State('graph1', 'figure'), State('input_annotations', 'value')])
def update_graph(n_clicks, figure, annotation):
    if n_clicks and annotation:
        # Check if 'annotations' key exists in the layout
        if 'annotations' not in figure['layout']:
```

```
            figure['layout']['annotations'] = []

        figure['layout']['annotations'].append({
            'x': 8, 'y': 8, 'text': annotation,
            'showarrow': True, 'arrowhead': 1
        })
    return figure

# Run server
if __name__ == '__main__':
    app.run_server(debug=True)
```

3.11. Implementación de marcas de agua

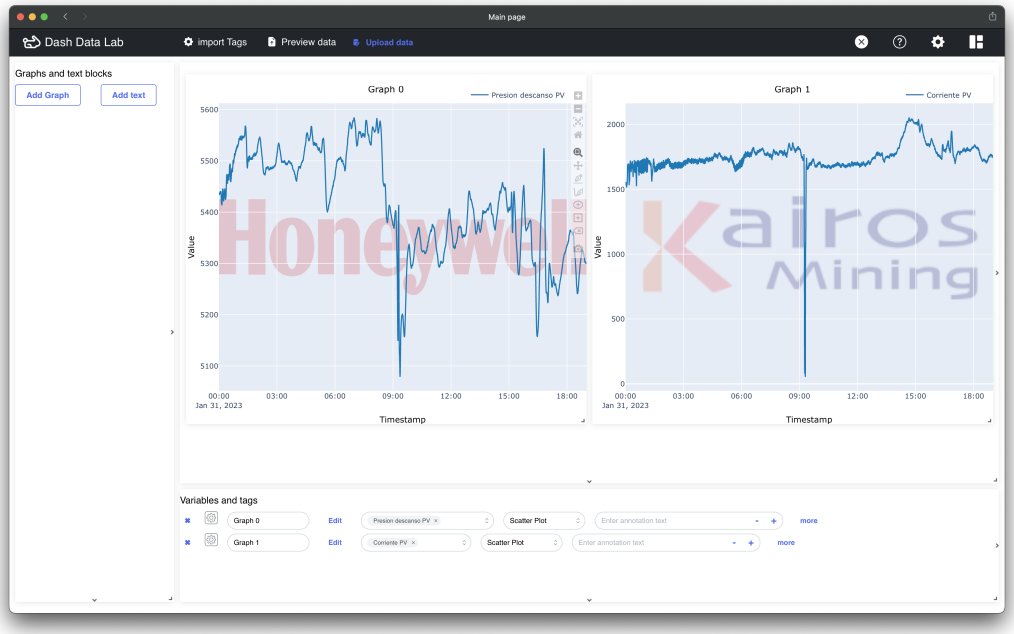


Figura 3.32: Ejemplos de marcas de agua

Las marcas de agua son una herramienta que nos permite disminuir la apropiación de propiedad intelectual por otras organizaciones o equipos dentro de la misma organización.

En el extracto presentado a continuación, se muestra cómo se puede agregar fácilmente

una marca de agua a una Figura a través del método `add_layout_image`, en el cual podemos configurar la imagen de fondo para que no obstruya la data que muestra la gráfica, como se muestra en la Figura 3.32.

```
    figure = go.Figure(data=go.Scatter(x=[0,2,3,5,7], y=[0,2,3,5,7]))
    figure.add_layout_image(
    dict(
        # Replace with your watermark image URL
        source='https://upload.wikimedia.org/wikipedia/commons/6/6b/Taka_Shiba.jpg',
        xref='paper', yref='paper',
        x=0.5, y=0.5,  # Position of the watermark
        sizex=1, sizey=1,  # Size of the watermark
        xanchor='center', yanchor='middle',
        opacity=0.2  # Watermark opacity
    )
)
```

4. Validación y resultados

En este capítulo se realizará un proceso de validación de la aplicación. Para ello se realizó un estudio de usabilidad sobre la aplicación desarrollada.

4.1. Pruebas de usabilidad

Con el objeto de medir el nivel de usabilidad de la aplicación desarrollada, se llevó a cabo un estudio utilizando el método User Testing [29]. De esta manera es posible identificar las falencias y fortalezas de su diseño, con el proposito de resolver interrogantes relevantes, tales como: ¿El usuario logra encontrar una determinada funcionalidad? o ¿Es una tarea fácil de lograr?

Para esto, se encomendó la realización de una serie de tareas o actividades a usuarios experimentados, pues queremos medir usabilidad y no aprendizaje en el sistema operativo Windows, navegadores web o análisis de datos.

El estudio se realizó a 5 personas, según el paper Determining Usability Test Sample Size [30] la mayor cantidad de problemas de usabilidad son detectados con las primeras 3 a 5 personas. Si se siguen realizando pruebas con más personas, es improbable encontrar nuevos problemas de usabilidad.

4.1.1. Descripción del estudio

Las tareas que se evaluaron son:

1. Crear un cuadro de texto con la palabras: "Información importante".
2. Cargar datos como archivo .csv o .xlsx desde el computador.
3. Buscar la variable "TEST_TAG" y extraer los datos de 1 hora hacia atrás desde ahora.
4. Crear un scatter plot a partir de los datos cargados.

5. Agregar 2 gráficos al área de trabajo y cambiar el tamaño y posición de ambos.
6. Hacer anotaciones con dibujos y texto en el gráfico creado.
7. Agregar una marca de agua al gráfico creado.
8. Exportar un gráfico a imagen PNG

Estas tareas se registraron mediante un audio del usuario y un vídeo de la pantalla de la aplicación, incentivando a que los usuarios piensen en voz alta, con explícito consentimiento para realizar las grabaciones. Las pruebas se realizaron en un servidor de prueba Windows a través del protocolo de escritorio remoto [31].

Los aspectos de usabilidad a medir son los siguientes:

- ¿Las opciones que se presentan son fáciles de comprender?
- ¿Es alguna de las tareas difícil de lograr?
- ¿Existe en algún momento una barrera para los usuarios?
- ¿Los usuarios sienten frustración o incomodidad en algún momento?

Para cuantificar esto se guardaron los siguientes datos:

- Tasa de éxito de las tareas.
- Tiempo que toma realizar cada tarea.
- Valoraciones subjetivas de satisfacción, frustración y confianza.
- Comentarios de problemas encontrados.

Los usuarios están caracterizados de la siguiente manera:

- Usuarios mayores de 25 años.
- Con experiencia en el sistema operativo Windows y navegadores web.
- Conocimientos en el análisis de datos (Específicamente de la industria minera).

- No han usado nunca la herramienta.

Los usuarios de prueba fueron reclutados dentro de la organización Honeywell.

4.1.2. Resultados del estudio

En la tabla 4.1 se presentan los resultados promediados del estudio, cabe destacar que los resultados detallados se encuentran en la sección de 6, referida a los Anexos.

Tarea	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Completó sin ayuda	5	5	4	5	5	3	5	4
Tiempo promedio [s]	23.6	60	42.2	68	32.8	49.8	24.4	80.4

Tabla 4.1: Resultados promediados de los 5 usuarios de prueba.

De los resultados se extrae que todas las tareas se completaron exitosamente, pero algunos usuarios presentaron problemas para encontrar algunas opciones, por lo que requirieron ayuda adicional.

4.2. Análisis de usabilidad

Tareas Evaluadas y Observaciones

- **¿Las opciones que se presentan son fáciles de comprender?**

  Las opciones no siempre fueron fáciles de comprender para los usuarios. Hubo varias instancias donde los usuarios tuvieron dificultades en encontrar opciones específicas, lo que indica que aunque la tarea fue completada, la comprensión de las opciones presentadas no fue inmediata o intuitiva para todos los usuarios.

- **¿Es alguna de las tareas difícil de lograr?**

  Las tareas, en general, fueron completadas con éxito por los usuarios. Sin embargo, la variabilidad en los tiempos de ejecución y la necesidad de asistencia adicional en algunas tareas, como la adición de una marca de agua a los gráficos, sugiere

que ciertas tareas presentaron dificultades. Esto no necesariamente indica una alta dificultad, pero sí señala áreas donde la usabilidad podría mejorarse.

- **¿Existe en algún momento una barrera para los usuarios?**

  Aunque no se menciona explícitamente una "barrera" en los resultados, las dificultades encontradas por los usuarios al buscar opciones específicas y la solicitud de ayuda adicional en algunas tareas sugieren que existieron barreras momentáneas en la usabilidad. Estas barreras no impidieron completar las tareas, pero sí representaron obstáculos que afectaron la fluidez de la experiencia del usuario.

- **¿Los usuarios sienten frustración o incomodidad en algún momento?**

  La necesidad de asistencia adicional y las dificultades encontradas al buscar opciones causaron pequeños momentos de frustración, pero en general los usuarios se sentían con la confianza de encontrar todo lo que se necesitaba con la información provista.

En resumen, dado que los usuarios pudieron completar las tareas asignadas, de momento la usabilidad de la herramienta se considera como un área positiva y un objetivo logrado. Por otra parte las dificultades encontradas sugieren áreas de mejora en la claridad y accesibilidad de las opciones, así como en la intuitividad general de la interfaz de usuario. Estos factores pueden influir en la experiencia general del usuario y deberían ser considerados para futuras mejoras de la aplicación.

4.3. Pruebas de estrés

Con el objetivo de determinar los límites de capacidad de la herramienta, se diseñaron seis archivos de prueba con cantidades de datos en aumento. Estos archivos varían en tamaño, empezando con 5 mil filas y escalando hasta 1 millón de filas. Cada uno contiene 16 columnas, lo que implica que el archivo más pequeño tiene 80 mil celdas de datos, más una columna para el timestamp, y el más grande llega a 16 millones de celdas, incluyendo también una columna de timestamp de 1 millón de filas.

La prueba comenzó con el archivo de 5 mil filas, una cantidad manejable conocida para la herramienta, proporcionando así una base sólida para las pruebas subsiguientes.

Para evaluar el rendimiento de la aplicación, se realizaron las pruebas siguientes:

- Cargar datos en la herramienta.
- Añadir una gráfica nueva.
- Eliminar una gráfica existente.
- Añadir un trazo a una gráfica, es decir, graficar una variable.
- Retirar un trazo de la gráfica.

Cada una de estas pruebas se repitió diez veces con cada tamaño de archivo. Se cronometró el tiempo desde el inicio de cada tarea hasta que fue posible realizar una nueva acción.

Los resultados detallados se presentan a continuación.

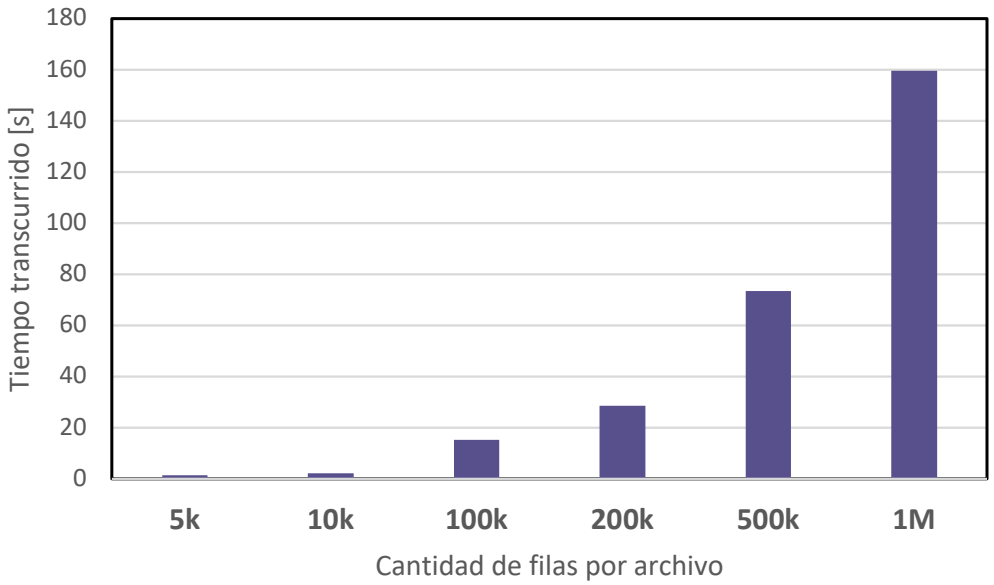


Figura 4.33: Test de carga de datos.

Como se ve en los resultados de la figura 4.33, a medida que el archivo aumenta de tamaño, el tiempo de cargar esos datos aumenta casi linealmente, ya que observa que para 10k filas se demora aproximadamente 1.5[s], para 100k filas demora 10 veces más y para 1 millón de datos 10 veces más.

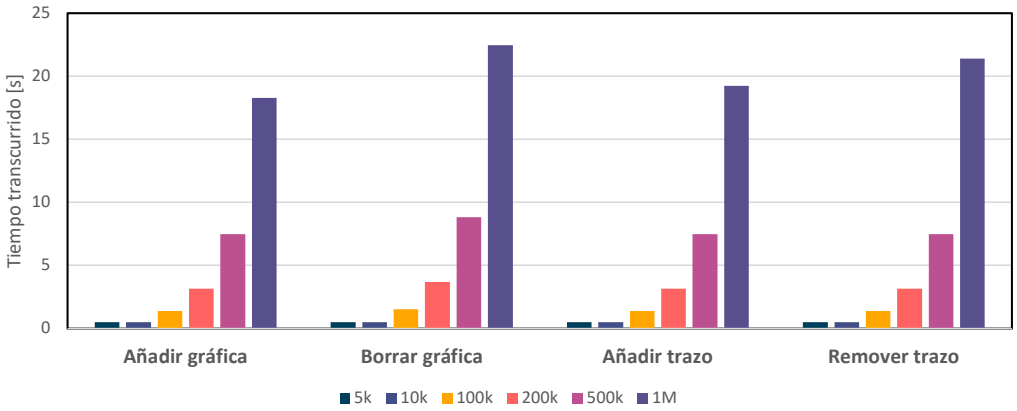


Figura 4.34: Múltiples pruebas de gráficas.

En la figura 4.34 podemos ver casi los mismos resultados que para cargar datos pero

tiempos mucho más cortos incluso al cargar 1 millón de filas. A simple vista se podría observar que las acciones de agregar son un poco menos demorosas que las de quitar algo, pero eso solo se ve al cargar el máximo de datos y en la práctica no es perceptible.

Si comparamos todas las pruebas (4.35) podemos ver que la tarea más demorosa es la de cargar los datos, ya que una vez cargados los datos, las otras tareas requieren mucho menos tiempo para ser ejecutadas.

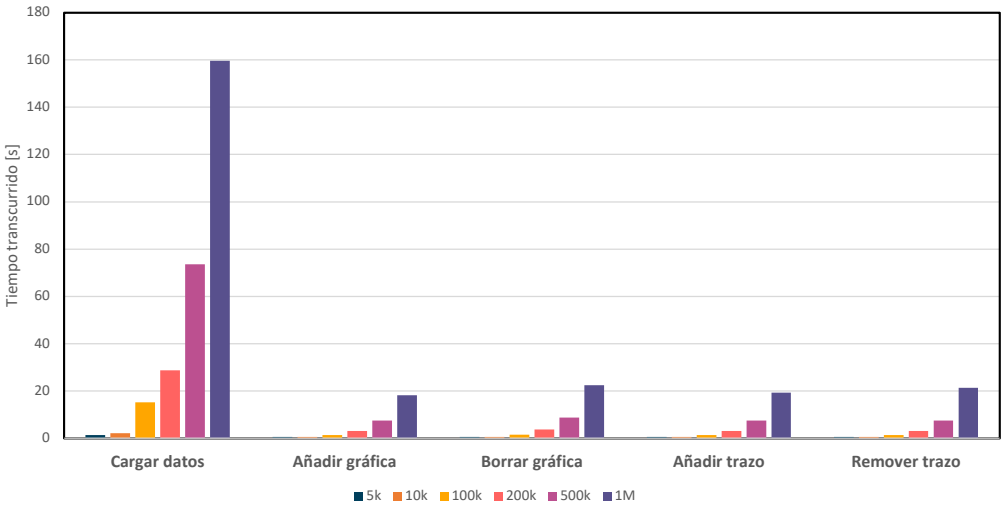


Figura 4.35: Todas las pruebas realizadas.

De los resultados obtenidos se puede concluir que la herramienta se mantiene relativamente fluida hasta las 500k filas (8 millones de datos), ya que el tiempo de carga es corto en comparación con la siguiente prueba y el tiempo de realizar el resto de las operaciones es manejable para un usuario.

Cabe destacar que incluso al cargar un archivo de 16 millones de datos, la herramienta se mantiene completamente utilizable y nunca se detiene el proceso.

5. Conclusiones

Este documento describe el desarrollo de una aplicación enfocada en la integración de la extracción, procesamiento, análisis y visualización de datos en sistemas de control avanzado, particularmente en la industria minera. A través de una interfaz gráfica intuitiva y la integración de tecnologías como Dash/Plotly, Python y MongoDB, la herramienta ofrece soluciones eficientes y personalizables para el manejo de datos confidenciales y la toma de decisiones estratégicas.

Se ha demostrado cómo esta herramienta no solo resuelve desafíos actuales, sino que también se adapta a necesidades futuras, enfatizando la seguridad de los datos y la adaptabilidad. Su capacidad para generar reportes y visualizaciones interactivas facilita la mejora continua de los procesos y la eficiencia operativa.

Para futuros desarrollos se recomienda incluir capacidades de detección automática de eventos y la expansión de la herramienta para abordar necesidades específicas del sector minero. Estudios de caso adicionales podrían servir para validar y perfeccionar aún más la herramienta según los requisitos de la industria minera.

En resumen, este trabajo representa un avance importante en el análisis de datos para la industria minera, ofreciendo una solución robusta, escalable y fácil de utilizar que mejora la eficiencia y apoya la toma de decisiones informadas.

5.1. Trabajo futuro.

Con base en los resultados obtenidos a lo largo del trabajo, se proponen las siguientes posibles mejoras del trabajo presentado en el trabajo de memoria, los cuales pueden llevarse a cabo en un futuro:

-

Referencias

[1] D. Plotly, "Dash," https://dash.plotly.com, accessed: 2023-12-30.

[2] Microsoft, "Microsoft paint," https://www.microsoft.com/en-us/windows/paint, accessed: 2023-12-30.

[3] M. P. BI, "Power platform products," https://www.microsoft.com/en-us/power-platform/products/power-bi, accessed: 2023-12-30.

[4] Seeq, "Advanced analytics for process manufacturing data," https://www.seeq.com, accessed: 2023-12-30.

[5] Streamlit, "The fastest way to build and share data apps," https://streamlit.io, accessed: 2023-12-30.

[6] "Python," https://www.python.org, accessed: 2023-12-30.

[7] Plotly, "Graphing library," https://plotly.com/, accessed: 2023-12-30.

[8] Microsoft, "Windows 8 system requirements," https://support.microsoft.com/en-us/windows/system-requirements-2f327e5a-2bae-4011-8848-58180a4353a7, accessed: 2023-12-30.

[9] ——, "Windows server 2012 r2," https://www.microsoft.com/es-es/evalcenter/evaluate-windows-server-2012-r2, accessed: 2023-12-30.

[10] Honeywell, "Uniformance phd," https://pmt.honeywell.com/learn/c/uniformance-phd?x=6yizg0, accessed: 2023-12-30.

[11] M. Atlas, "Cloud database service," https://www.mongodb.com/atlas/database?tck=docs_server, accessed: 2023-12-30.

[12] Microsoft, "File formats supported in excel," https://support.microsoft.com/en-au/office/file-formats-that-are-supported-in-excel-0943ff2c-6014-4e8d-aaea-b83d51d46247, accessed: 2023-12-30.

[13] "Creativyst csv: Comma separated value file format," https://www.creativyst.com/Doc/Articles/CSV/CSV01.shtml#FileFormat, accessed: 2023-12-30.

[14] W. W. H. A. T. W. Group), "Html living standard," https://html.spec.whatwg.org/multipage/, accessed: 2023-12-30.

[15] React, "React - a javascript library for building user interfaces," https://react.dev, accessed: 2023-12-30.

[16] Pallets, "Flask - a lightweight wsgi web application framework," https://flask.palletsprojects.com/, accessed: 2023-12-30.

[17] "Dash bootstrap components," https://dash-bootstrap-components.opensource.faculty.ai, accessed: 2023-12-30.

[18] "Dash mantine components," https://www.dash-mantine-components.com/, accessed: 2023-12-30.

[19] D. Draggable, "Documentation," https://dash-draggable.readthedocs.io/en/latest/, accessed: 2023-12-30.

[20] Microsoft, "Dynamic link library troubleshooting," https://learn.microsoft.com/en-us/troubleshoot/windows-client/deployment/dynamic-link-library, accessed: 2023-12-30.

[21] D. Plotly, "Basic callbacks," https://dash.plotly.com/basic-callbacks, accessed: 2023-12-30.

[22] PythonNet, "Github repository," https://github.com/pythonnet/pythonnet, accessed: 2023-12-30.

[23] JetBrains, "dotpeek: Free .net decompiler & assembly browser," https://www.jetbrains.com/decompiler/, accessed: 2023-12-30.

[24] Millejoh, "Gist on github," https://gist.github.com/millejoh/e5eb0ccb409c1c5f201f1efd7ca466d3, accessed: 2023-12-30.

[25] pandas, "What is a dataframe?" https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html, accessed: 2023-12-30.

[26] MongoDB, "Pymongo documentation," https://www.mongodb.com/docs/drivers/pymongo/, accessed: 2023-12-30.

[27] "Markdown guide," https://www.markdownguide.org, accessed: 2023-12-30.

[28] A. Inc., "macos ventura," https://apps.apple.com/es/app/macos-ventura/id1638787999?mt=12, accessed: 2023-12-30.

[29] Usability.gov, "Usability testing," https://www.usability.gov/how-to-and-tools/methods/usability-testing.html, accessed: 2023-12-30.

[30] C. W. Turner, J. R. Lewis, and J. Nielsen, "Determining usability test sample size," 2006.

[31] Microsoft, "How to use remote desktop," https://support.microsoft.com/en-us/windows/how-to-use-remote-desktop-5fe128d5-8fb1-7a23-3b8a-41e636865e8c, accessed: 2023-12-30.

6. Anexos

6.1. Resultados detallados del estudio de usabilidad.

Tarea	Completada	Duración [s]	Observaciones
1	Si	21	
2	Si	53	
3	Si	27	El explorador de variables es similar al de PSES (Profit Suite Engineering Studio) y UPS (Uniformance Process Studio).
4	Si	41	
5	Si	35	
6	Si	42	Tuve dificultades para encontrar el menú, sin embargo, posteriormente resulta intuitivo.
7	Si	17	
8	Si	58	Tuve dificultades leves para encontrar la opción.

Tabla 6.2: Resultados usuario de prueba 1

Tarea	Completada	Duración [s]	Observaciones
1	Si	31	
2	Si	74	
3	Si	52	El TAG Browser es similar al de PI System.
4	Si	91	No encontraba la opción para hacer el grafico.
5	Si	33	
6	Si	86	Necesite ayuda para encontrar el menú. Es bastante fácil de utilizar una vez localizado el menú.
7	Si	24	
8	Si	107	

Tabla 6.3: Resultados usuario de prueba 2

Tarea	Completada	Duración [s]	Observaciones
1	Si	24	
2	Si	65	
3	Si	55	
4	Si	79	
5	Si	32	
6	Si	60	
7	Si	31	
8	Si	55	

Tabla 6.4: Resultados usuario de prueba 3

Tarea	Completada	Duración [s]	Observaciones
1	Si	20	
2	Si	65	
3	Si	50	
4	Si	88	
5	Si	32	Es muy intuitivo el cambiar de tamaño y posición los gráficos.
6	Si	30	
7	Si	26	
8	Si	72	

Tabla 6.5: Resultados usuario de prueba 4

Tarea	Completada	Duración [s]	Observaciones
1	Si	22	
2	Si	43	El arrastrar los archivos sobre la aplicación facilita considerablemente la actividad.
3	Si	27	
4	Si	41	
5	Si	32	
6	Si	31	
7	Si	24	
8	Si	110	Pidió ayuda para encontrar la opción

Tabla 6.6: Resultados usuario de prueba 5