



República Bolivariana de Venezuela
Universidad Nacional Experimental Politécnica
“Antonio José de Sucre”
Vice Rectorado Barquisimeto
Departamento de Ingeniería Electrónica



Herramienta computacional para el análisis de la vibración en motores eléctricos alimentada mediante datos de una simulación digital

Integrantes:
Gerardo Alfonzo Campos Fonseca
V. 27085179
José Andrés Cortez Teran
V. 26540824

Tutor: Dra. Luisa Escalona
Cotutor: Dr. Carlos Zambrano

Barquisimeto, Marzo del 2022



República Bolivariana de Venezuela
Universidad Nacional Experimental Politécnica
“Antonio José de Sucre”
Vice Rectorado Barquisimeto
Departamento de Ingeniería Electrónica



Herramienta computacional para el análisis de la vibración en motores eléctricos alimentada mediante datos de una simulación digital

Integrantes:

Gerardo Alfonzo Campos Fonseca

V. 27085179

José Andrés Cortez Teran

V. 26540824

Tutor: Dra. Luisa Escalona

Cotutor: Dr. Carlos Zambrano

“Trabajo Especial presentado ante el Departamento de Ingeniería
Electrónica de la Universidad Nacional Experimental Politécnica
“Antonio José de Sucre” Vicerrectorado de Barquisimeto como requisito
parcial para optar al título de Ingeniero Electrónico”

Barquisimeto, Marzo del 2022

Índice

Índice	III
Índice de figuras	VIII
Índice de Tablas	X
RESUMEN	XI
1. PLANTEAMIENTO DEL PROBLEMA	5
1.1. DESCRIPCIÓN DEL PROBLEMA	5
1.2. OBJETIVOS DE LA INVESTIGACIÓN	7
1.2.1. Objetivo general	7
1.2.2. Objetivos específicos	8
1.3. JUSTIFICACIÓN E IMPORTANCIA	8
1.4. LIMITACIONES Y ALCANCES	9
1.4.1. Limitaciones	9
2. REVISIÓN BIBLIOGRÁFICA	11
2.1. ANTECEDENTES	11
2.2. MARCO TEÓRICO	12
2.2.1. Motores eléctricos	12
2.2.2. Análisis de Vibración	14

2.2.3. Acelerómetros	16
2.2.4. Procesamiento de señales	18
2.2.5. Herramienta Computacional	20
2.2.6. Sistema Web	20
2.2.7. Base de datos (BBDD)	22
2.2.8. MongoDB	23
2.2.9. MongoDB Atlas	24
2.2.10. Interfaz de programación de aplicaciones (API)	24
2.2.11. Modelo estadístico	25
2.2.12. Prueba de bondad de ajuste	25
2.2.13. Prueba de Kolmogorov-Smirnov	25
2.2.14. Lenguaje de programación	26
2.2.15. Go	27
2.2.16. Fyne	27
2.2.17. HTML	28
2.2.18. CSS	28
2.2.19. JavaScript	29
2.2.20. React	30
2.2.21. Python	30
2.2.22. Django	31
2.2.23. Numpy	31

2.2.24. Pandas	32
2.2.25. SciPy	32
2.2.26. FastApi	32
2.2.27. Octave	33
2.2.28. Git	33
3. MARCO METODOLÓGICO	35
3.1. NATURALEZA DE LA INVESTIGACIÓN	35
3.2. TIPO DE INVESTIGACIÓN	35
3.3. FASES DE LA INVESTIGACIÓN	36
3.3.1. Definición de los requerimientos de usuario	36
3.3.2. Revisión documental	37
3.3.3. Análisis de la información	38
3.3.4. Diseño del modelo estadístico	38
3.3.5. Diseño de las vistas para mostrar la información	39
3.3.6. Implementación	39
3.3.7. Comprobar los resultados	42
3.4. METODOLOGÍA	43
3.4.1. Metodología de desarrollo de software	43
4. ANÁLISIS Y DISCUSIÓN DE RESULTADOS	46
4.1. PROCEDIMIENTO DE RECOLECCIÓN DE INFORMACIÓN	46

4.2. ELECCIÓN DE LAS HERRAMIENTAS Y LENGUAJES	47
4.2.1. Servidor	47
4.2.2. Cliente	48
4.2.3. Modelo estadístico	49
4.2.4. Análisis en frecuencia	50
4.3. IMPLEMENTACIÓN DEL MODELO ESTADÍSTICO	51
4.4. IMPLEMENTACIÓN DEL ANÁLISIS EN FRECUENCIA	53
4.5. IMPLEMENTACIÓN DE LA BASE DE DATOS	54
4.6. LLENADO DE LA BASE DE DATOS	55
4.7. IMPLEMENTACIÓN DE LOS SERVIDORES	56
4.7.1. Servidor dedicado a Sensorica	57
4.7.2. Servidor Web	60
4.8. IMPLEMENTACIÓN DE LOS CLIENTES	63
4.8.1. Cliente de Sensorica	63
4.8.2. Cliente Web	65
4.9. COMPROBACIÓN DE LOS RESULTADOS	67
4.9.1. Comprobación de resultados a nivel técnico	68
5. CONCLUSIONES Y RECOMENDACIONES	73
5.1. Conclusiones	73
5.2. Recomendaciones	76

<i>ÍNDICE</i>	VII
ANEXOS	84
5.3. ANEXO A. Manual de Uso e Instalación	85
5.3.1. Vista General	86
5.3.2. Vista Especifica	87
5.3.3. Vista Exhaustiva	89

Índice de figuras

1.	Diagrama de los dominios temporal y frecuencial	19
2.	Base de datos MongoDB Atlas	56
3.	Llenado de la BBDD	58
4.	Servidor de sensorica	60
5.	Servidor Web	63
6.	GUI del cliente de sensorica	66
7.	Configuración de proceso en Linux para reinicio automático de procesos	68
8.	Prueba de error, no unicidad de información	69
9.	Prueba de error, no BBDD Sensorica	70
10.	Prueba de error, no BBDD Web	70
11.	Prueba de error, petición no completada en el cliente Web	71
12.	Prueba de error, petición exhaustiva inválida	71
13.	Header de la herramienta computacional.	85
14.	Footer de la herramienta computacional.	85
15.	Vista de la pestaña general de la herramienta computacional.	86
16.	Controles de paginación en la vista principal.	87
17.	Sección de mensaje en la vista especifica.	87
18.	Sección de gráficas en la vista especifica.	88
19.	Sección de la tabla en la vista especifica.	89

20.	Sección de mensaje en la vista exhaustiva.	89
21.	Sección de gráfica de frecuencia en la vista exhaustiva.	90

Índice de Tablas

1.	Comparativa de posibles lenguajes nivel servidor	48
2.	Comparativa de posibles lenguajes nivel cliente Web	49
3.	Comparativa de posibles lenguajes para el análisis exploratorio	49
4.	Comparativa de posibles lenguajes para el análisis de frecuencia . . .	50
5.	Estructura de MotorData	54
6.	Estructura de MotorInDB	55
7.	Estructura IdSensor	55
8.	Banderas Script para el llenado de la BBDD	57
9.	Funciones Servidor Sensorica	59
10.	Funciones Servidor Web	61



República Bolivariana de Venezuela
Universidad Nacional Experimental Politécnica
“Antonio José de Sucre”
Vice Rectorado Barquisimeto
Departamento de Ingeniería Electrónica



Herramienta computacional para el análisis de la vibración en motores eléctricos alimentada mediante datos de una simulación digital

Autores: Gerardo Campos, José Cortez

Tutor: Dra. Luisa Escalona, **Cotutor:** Dr. Carlos Zambrano

La presente investigación ha tenido como propósito el desarrollo de una herramienta computacional para el análisis de la vibración en motores eléctricos, haciendo uso de datos obtenidos mediante una simulación digital para verificar su funcionamiento y una metodología de desarrollo Web enfocada a microservicios. Para lograrlo se desarrolló un modelo estadístico, capaz de entregar las mediciones necesarias para la implementación, a partir de una base de datos existente; así mismo se crearon 3 microservicio para cubrir las necesidades de obtención de información y comunicación con el modelo, almacenar la información (database as a service DBaaS) y mostrar la información por medio de una página Web; se desarrollaron estructuras de datos y se diseñó una base de datos no relacional para el almacenamiento de la información, se automatizó la interconexión entre el modelo estadístico, el servidor de adquisición de información y el DBaaS mediante un cliente y un Script y se despliega todo el servicio en una página Web con 3 vistas, General, específica, exhaustiva, en los cuales se puede obtener una vista global del estado de los motores en la base de datos, un recuento de la evolución histórica del motor (por medio de gráficas y una tabla) y una gráfica de un análisis en frecuencia correspondiente al espectro de vibración que posee el motor; además la página consta de un manual de usuario para facilitar su utilización. De esta forma de tiene un sistema completamente modular que puede escalar con el tiempo y puede ser alimentado por cualquier sistema de adquisición de datos que respete los formatos y estructuras definidas.

Palabras claves: servidor, microservicio, modelo estadístico, vibración, motores eléctricos.

CAPÍTULO 1

PLANTEAMIENTO DEL PROBLEMA

1.1. DESCRIPCIÓN DEL PROBLEMA

En la actualidad se vive un crecimiento exponencial a nivel industrial en todos los rubros, desde manufactura hasta desarrollo de ciencias aplicadas, especialmente existe una muy alta demanda de alimentos, la evolución y la posibilidad de satisfacer tal demanda es debido, entre otras cosas, al motor eléctrico, este es un artefacto que transforma la energía eléctrica en energía mecánica (movimiento), de manera que puede impulsar el funcionamiento de una máquina y son utilizados ampliamente en: Bombas para Agua, Bombas Industriales, Mezcladoras, Molinos, Correas Transportadoras, Zarandas, Cortadoras, Ventiladores, Grúas, y en todo proceso que involucre movimiento.

Adicional a la versatilidad de uso de los motores, existen factores que deben ser considerados al evaluar su rendimiento, como las grandes pérdidas (horas, insumos, dinero, etc.) que ocasiona una parada de emergencia en una planta, por lo cual se considera de vital importancia que los motores se encuentren completamente operativos y funcionales. Para procurar su buen estado y funcionamiento, se deben realizar mantenimientos.

Existen diferentes tipos de mantenimiento, entre ellos están:

- Correctivo: se espera que ocurra una falla para reparar o cambiar un equipo. Esto puede degradar la vida útil del equipo y debido a que la falla puede ocurrir en cualquier momento, lo que usualmente produce un paro en la línea de producción; por lo tanto, este tipo de mantenimiento suele y debe ser evitado.
- Preventivo: para evitar una falla mayor se detiene la maquinaria para hacer un mantenimiento preparado con anticipación, se inspecciona la maquinaria y

se reemplazan las piezas propensas a dañarse. Este tipo de mantenimiento en algunas circunstancias es más que suficiente pero en el caso de los rodamientos puede ser contraproducente.

- Predictivo: se pronostica cuando una falla está a punto de ocurrir; a través de mediciones, y del estudio de las mismas, se prevé cuando un desperfecto está a punto de ocurrir y de esta forma se realiza una mejor planificación. Cabe resaltar que este tipo de mantenimiento puede dar información acerca del origen y la gravedad de las averías.

En el caso de los motores eléctricos, como dice Lacey (s.f.), el mantenimiento preventivo tiene muchas desventajas dado que existen problemas de índole mecánico así como administrativos y monetarios, un ejemplo de esto pueden ser los altos costos de reemplazo dado que las partes se reemplazan muy pronto, el riesgo de pérdida completa o parcial dado un error humano, como la posibilidad de generar daño o una incorrecta instalación de la misma, la instalación de una pieza defectuosa, y por último, el hecho de que las piezas reemplazadas pueden tener muchos años de vida útil.

De esta forma se ve que una observación constante ofrece más control sobre las variables mencionadas y, aunque no evita las posibilidades de error humano, si permite reaccionar a este; adicionalmente, se deben considerar las altas pérdidas y retrasos, sumadas a las dificultades administrativas, que generan las paradas periódicas de la planta. Todas estas consideraciones son de suma importancia ya que para poder ser ejecutadas se necesitan herramientas capaces de dar a conocer el estado actual de una máquina y permitir a los operadores, encargados o ingenieros tomar acciones cuando sean necesarias y de acuerdo a las políticas de la empresa.

Por las razones expuestas surge la necesidad de reconocer las fallas y de tener bajo continuo monitoreo los factores que las causan para, de esta forma, atenuarlos. Según los estudios realizados por Kammermann y col. (2017) la media de la probabilidad de fallas en máquinas de inducción permanece al nivel de los años 70 ($10^{-6}/hour$) y está altamente relacionada a la falla de los rodamientos, y que un 59% de las fallas son causadas por los rodamientos, esto es debido a que son piezas sometidas a

mucho estrés mecánico, permiten soporte y asimismo necesitan tener poca fricción. Por esta razón, su principal falla es el desgaste, de igual forma se presentan otras fallas estructurales que generan vibraciones. Cabe resaltar que todas las fallas mecánicas generan vibración sin importar su relación con los rodamientos y lo hacen a distinta frecuencia (f).

Por lo expuesto, es de suma importancia el estudio de la vibración, para la que se suele implementar un acelerómetro y con un estudio de la frecuencia se puede obtener la causa y la magnitud de la falla y de esta forma programar su reparación. Sin embargo, debido a que esta evaluación debe ser realizada en cada rodamiento y acople o extensión del motor y dadas las mediciones que se deben realizar por unidad (motor y acoples) y por la gran cantidad de unidades existentes a nivel industrial, es virtualmente imposible el estudio con un acelerómetro convencional (a pesar de que la medición no sea un proceso muy largo y los cálculos y evaluaciones sean realizados posteriormente). Sumado a esto, las industrias suelen poseer medidas y controles sanitarios, aumentados por la pandemia actual, que no permiten el constante monitoreo de la planta significando esto la imposibilidad de implementar este tipo de acciones de forma manual, por lo cual se debe recurrir a un sistema de automatización capaz de medir las vibraciones en todos estos equipos que, a su vez, permita el estudio de estos datos de forma remota.

1.2. OBJETIVOS DE LA INVESTIGACIÓN

1.2.1. Objetivo general

Desarrollar una herramienta computacional para el análisis de la vibración en motores eléctricos, mediante la simulación digital de un acelerómetro, con la finalidad de que un operador determine averías y sus causas.

1.2.2. Objetivos específicos

1. Justificar la escogencia de las herramientas y lenguajes a utilizar en las diferentes etapas que requiere la simulación. (José Cortez y Gerardo Campos)
2. Generar un modelo estadístico de la vibración en motores eléctricos con distinto grado de daño utilizando una base de datos de la salida de un acelerómetro digital. (José Cortez)
3. Elaborar una base de datos con información obtenida del modelo estadístico para alimentar los niveles de análisis de la herramienta. (José Cortez y Gerardo Campos)
4. Realizar análisis de fallas en frecuencia, a partir de la salida del modelo del acelerómetro. (Gerardo Campos)
5. Mostrar la información solicitada de acuerdo al nivel de análisis seleccionado. según sea: Vista Principal, Vista Específica o Vista Exhaustiva. (José Cortez y Gerardo Campos)
6. Elaboración de una página Web para facilitar la utilización del sistema (José Cortez y Gerardo Campos)
7. Comprobar los resultados de la herramienta de análisis. (José Cortez y Gerardo Campos)

1.3. JUSTIFICACIÓN E IMPORTANCIA

Para prevenir las fallas mecánicas que ocurren en los motores, por el deterioro y desgaste de los rodamientos, estos se deben tener bajo constante monitoreo para poder efectuar un mantenimiento puntual que elimine dichos peligros. De esta forma el mantenimiento predictivo es la clave para mejorar la vida útil, funcionamiento y planificación de todo proceso especialmente a niveles industriales, donde la cantidad de motores eléctricos es bastante elevada por lo cual la automatización del proceso es crucial.

Sin embargo, dado los altos costos que implican realizar una automatización y en especial a escalas industriales se suele hacer una simulación o una emulación para estudiar y obtener el modelo más preciso, gracias a la facilidad de manipular con mayor eficacia los diseños y conocer su comportamiento real sin necesidad de construirlo, antes de realizar la implementación y todo el proceso que esta conlleva.

Habiendo expuesto la importancia del mantenimiento como también la de realizar simulaciones, se justifica el hecho de realizar el desarrollo del trabajo aquí propuesto el cual permita emular el comportamiento y las salidas de un acelerómetro, como también otorgar las herramientas necesarias para realizar un mantenimiento predictivo, y de esta forma estudiar a profundidad su estado actual como también su evolución histórica. Todo esto sumado a las facilidades de portabilidad que ofrece un sistema Web, facilitando la revisión constante sin las dificultades de los protocolos de acceso y sanidad usuales en las plantas industriales.

1.4. LIMITACIONES Y ALCANCES

En función de los objetivos planteados con anterioridad, se puede definir tanto las limitaciones como el alcance del proyecto.

1.4.1. Limitaciones

- Recursos económicos impiden la adquisición de dispositivos para pruebas en motores reales.
- Disponibilidad de muestras, aunque se cuenta con una base de datos lo suficiente grande para cubrir el comportamiento de la vibración en motores, incluso de distinta potencia, esta es discreta y con intervalos de tiempo considerables entre cada muestra.
- Las establecidas por el software de terceros utilizado en el diseño y desarrollo del proyecto.

1.4.2. Alcance

El principal alcance de este trabajo es el desarrollo de una herramienta computacional para el análisis de la vibración en motores eléctricos; en función de esto:

- La herramienta computacional permite una vista general del estado de todos los motores en un espacio previamente delimitado y seleccionado (planta o piso) que se encuentren en la base de datos.
- La herramienta computacional permite una vista detallada del estado de un motor seleccionado, previamente por el usuario, que se encuentre en la base de datos.
- El sistema cuenta con una base de datos a partir de la cual se desarrollarán los resultados ofrecidos.
- No se contempla la construcción de hardware de ningún tipo.

CAPÍTULO 2

REVISIÓN BIBLIOGRÁFICA

2.1. ANTECEDENTES

Çağlar Ramazan y col. (2014) Describen un modelo estadístico sobre el deterioro de motores eléctricos de inducción. El artículo clasifica el deterioro del motor en 7 etapas de acuerdo al nivel de vibración de sus rodamientos. Este antecedente es de gran utilidad para realizar el modelo estadístico que alimenta la base de datos ya que proporciona bases teóricas y un ejemplo práctico.

Pinto y col. (2016) realizan una simulación de redes de sensores inalámbricos, con el fin de mejorar la detección de fallas de los sensores. Esta simulación fue realizada con Castalia un simulador de redes de sensores inalámbricos y el sensor usado fue un detector de luminosidad. Algunas de las razones que menciona el artículo de por qué fue escogida una simulación es debido a que realizar el experimento con sensores requiere un costo elevado de hardware y un estudio analítico no es efectivo, ya que la complejidad del sistema es muy grande. Este trabajo es de utilidad debido a que sirve como orientación a la elaboración de la simulación del sensor que generaría el modelo estadístico.

Ugwiri y col. (2020) Presentan un resumen de las técnicas más usadas actualmente en la detección de fallas en motores eléctricos mediante análisis de vibración, además de realizar un experimento donde se ponen a prueba algunos de estos conceptos. El propósito de este antecedente es el de apoyar la elaboración del modo de "vista exhaustiva".

Koene y col. (2020) Elaboran un sensor de vibración inalámbrico de código libre llamado Memsio. Este dispositivo es alimentado por baterías y permite la adquisición de datos a alta velocidad por medio del uso de un acelerómetro microelectromecánico. Los autores mencionan que en la industria los sensores de aceleración más usados

son los piezoeléctricos, debido a tener una mayor precisión y tolerancia al ruido, sin embargo, los avances de los dispositivos microelectromecánico y su bajo costo hacen cada vez más factibles su uso para el monitoreo. Este antecedente muestra la tendencia de la reducción de precios de los sensores inalámbricos lo cual apoya al propósito del trabajo al hacer factible las redes de sensores.

Soto-Ocampo y col. (2020) Elaboraron un sensor de vibración multicanal para vibraciones de alta frecuencia utilizando una Raspberry pi. El objetivo del proyecto era conseguir una alternativa de bajo costo, para poder monitorizar la salud de rodamientos en motores eléctricos, pero sin sacrificar la calidad de la medición porque, como explican los autores del artículo, la mayoría de las fallas en los rodamientos ocurre a altas frecuencias y muchas de las alternativas de bajo costo no alcanzan la frecuencia necesaria. Este antecedente es de utilidad dada la cantidad de información, tanto teórica como datos de simulación, que provee.

2.2. MARCO TEÓRICO

Esta parte del capítulo expone el contenido teórico necesario para la realización de este proyecto. Esto incluye información sobre los motores eléctricos, análisis de la vibración, las herramientas computacionales, los sistemas y el modelado estadístico.

2.2.1. Motores eléctricos

De acuerdo a Mora (2003), un motor eléctrico es un dispositivo que transforma energía eléctrica en energía mecánica mediante la acción de campos magnéticos y están compuestos, principalmente, por un estator (parte fija) y un rotor (parte móvil). Existen dos familias principales de motores eléctricos las cuales, a su vez, se subdividen en **motores de corriente alterna** como lo son los motores de inducción, síncronas, entre otros y los **motores de corriente continua** como lo son los motores de escobillas, sin escobillas, de imán permanente, entre otros.

El motor más usado, en el sector industrial, es el motor de inducción debido

principalmente a su bajo costo y al poco mantenimiento que requiere para estar completamente operativo, esto en particular es debido a su sencillo diseño en comparación al de otros motores de igual potencia, ya que requiere menor número de componentes. Cabe resaltar que el segundo tipo de motor más usado es el motor de corriente continua, debido a que su control, en términos de torque y potencia, es mucho mas sencillo para ciertos niveles de potencia.

También existen los motores síncronos, cuya construcción es similar a la de un motor de inducción pero a su vez requiere de una fuente de alimentación externa para el rotor, lo cual aumenta su costo, además de esto su velocidad es constante, por lo tanto este tipo de motor es utilizado en aplicaciones específicas. Como se mencionó anteriormente, existen otros tipos de motores, pero son usualmente de menor potencia por lo tanto su utilidad industrial es mucho más limitada.

Rodamientos

Para que un motor pueda llevar a cabo la transformación de potencia debe rotar. Esta acción es llevada a cabo por el **rotor**, el cual esta formado por un eje que soporta un juego de bobinas envueltas sobre un núcleo magnético. Este se encuentra suspendido por **rodamientos**, de forma que el movimiento y las fuerzas producidas en la interacción entre las bobinas y el núcleo con los campos magnéticos, producidos por el estator, puedan ser utilizados. Además, los rodamientos, cuando se encuentran en buenas condiciones, permiten minimizar el roce entre el eje y soporte, maximizando la transferencia de potencia.

Estos también son conocidos como **rolineras**, las cuales según LTD (2021), son un tipo de cojinete, un elemento mecánico diseñado para reducir la fricción entre un eje y los elementos conectados al mismo. Existen muchos tipos de rolineras, sin embargo su estructura se fundamenta en dos anillos concéntricos, algún elemento rotativo como pueden ser bolas o rodillos, una jaula que se encarga de mantener los elementos de rodadura separados además de guiados y un lubricante o un sistema de lubricación.

Dado que son un elemento mecánico, el cual es continuamente usado, sufre mucho

desgaste y es el elemento más propenso a dañarse; según los estudios realizados por Kammermann y col. (2017) el 59% de las fallas son causadas por los rodamientos y asimismo su principal falla es el desgaste, de igual forma se presentan otras fallas estructurales. Por estas razones es fundamental tener bajo continuo monitoreo este elemento, esto se suele hacer a través de un análisis de vibración.

2.2.2. Análisis de Vibración

Como se explica en Sauer202 y col. (2021), la vibración es el movimiento periódico de un cuerpo o medio elástico, alrededor de un punto de equilibrio. En general podemos decir que una vibración es un caso específico de una oscilación, cuando esta es de origen mecánico.

Asimismo, se conoce como análisis de vibración, al conjunto de técnicas que permite obtener información de un equipo, a partir de sus vibraciones. Es una de las técnicas más usadas en el mantenimiento predictivo de equipos mecánicos, debido a que es un proceso poco invasivo y de bajo costo. El análisis de vibración permite diagnosticar fallas de forma temprana, así como detectar señales prematuras de desgaste.

Según Girdhar Girdhar y col. (2004), el análisis de vibración nos permite detectar las siguientes fallas:

- Defectos en los engranajes
- Defectos en los rodamientos
- Desalineamientos
- Desbalances
- Ejes torcidos
- Excentricidad
- Fallas eléctricas
- Fuerzas hidráulicas o aerodinámicas
- Mala sujeción en las piezas
- Problemas en las correas de transmisión
- Problemas de lubricación

- Resonancia
- Rozamientos en el rotor

Adquisición de datos

Para realizar cualquier tipo de análisis, primero se deben adquirir datos y, por regla general, mientras más datos se tengan y más precisas sean las mediciones, mejores serán los estudios que se pueden realizar.

La adquisición de datos, según Smith (2020), es el proceso de realizar mediciones de fenómenos físicos y registrarlos, en algún formato específico, para analizarlos posteriormente. Cabe resaltar que en algunos casos es necesario hacer un acondicionamiento de la señal, esto implica modificarla controladamente al reducir o aumentar su amplitud además de sumarle un nivel offset, voltaje continuo, para que cumpla requisitos mínimos y pueda ser procesada por los siguientes circuitos.

Al momento de la adquisición de datos, se toman medidas de una señal analógica la cual se convertirá al pasar por una serie de etapas y dispositivos en una señal digital y se guardará en un formato deseado y en una unidad de almacenamiento masivo (ROM, flash, etc.). El primer paso en la toma de datos comienza con el sensor, que es un dispositivo el cual transforma la unidad física de interés, en una señal que pueda ser procesada con mayor facilidad, luego se adecuará mediante un circuito especializado a las características y requerimientos del sistema, posteriormente será procesada por un convertidor Analógico-Digital, el cual se encarga de muestrear, retener y procesarla y, de esta forma, obtener una versión digital de la misma, la cual se guardará mediante un software desde una computadora.

En el caso de las vibraciones, los sensores más usados son los acelerómetros. Esto se debe principalmente a que tienen mayor ancho de banda que los sensores de posición y los de velocidad, lo que les permite detectar vibraciones de mayor frecuencia. Adicionalmente, dado que las vibraciones en los rodamientos, por naturaleza, son de alta frecuencia y, como se mencionó anteriormente, son componentes críticos en los motores eléctricos. Estas características hacen al acelerómetro el sensor más

utilizado para las mediciones de vibración en motores eléctricos. Sin embargo, en casos más especializados, como podría ser el análisis de vibración de maquinaria de larga envergadura, son también usados los otros tipos de sensores ya que sus características pueden facilitar el estudio.

2.2.3. Acelerómetros

Como se expone en Fraden (2003), un acelerómetro es un dispositivo capaz de medir aceleración, no es necesariamente la misma que la aceleración de coordenadas (cambio de la velocidad de un elemento en el espacio), sino que es la correlación asociada con el fenómeno de peso experimentado por una masa de prueba que se encuentra en el marco de referencia del dispositivo. Funciona mediante la utilización de la segunda ley de Newton, **la fuerza resultante ejercida sobre un cuerpo es proporcional a su masa por su aceleración**. Los acelerómetros por lo general cuentan con una masa de prueba (también conocida como masa sísmica), alguna especie de resorte y un marco de soporte que, a su vez, puede funcionar como amortiguador. Debido a esto, los acelerómetros se pueden modelar matemáticamente como un sistema lineal de segundo orden, por lo que su respuesta en frecuencia posee un pico de resonancia.

Tipos de acelerómetros

- Acelerómetros Capacitivos:

Los acelerómetros capacitivos son uno de los modelos más sencillos capaces de medir aceleración, además de ser fácilmente utilizables y reproducibles en masa. Funcionan mediante una masa sísmica dado que al experimentar una fuerza se desplaza con una aceleración proporcional a la fuerza aplicada. Si a la masa se le agregan unos resortes unidos a una carcasa, los resortes ejercerán una fuerza proporcional al desplazamiento de la masa generando un desplazamiento fácilmente medible.

Este fenómeno se produce ya que estos efectos en conjunto al amortiguamiento producen un sistema lineal de segundo orden, este sistema matemático genera

una salida en desplazamiento al aplicarse como entrada una fuerza Finalmente, al conectarse un sensor de desplazamiento, se observa la aceleración a la que esta sometido el instrumento. El sensor de desplazamiento más popular para este tipo de medidores es el capacitivo.

En su mayoría, los acelerómetros capacitivos vienen como dispositivos microelectromecánicos (**mems** por sus siglas en inglés) que son dispositivos fabricados con técnicas similares a las de fabricación de circuitos integrados; a su vez, poseen componentes mecánicos microscópicos. Son los acelerómetros más populares en aplicaciones no industriales, sin embargo, en aplicaciones de bajo consumo, bajo coste o cuando las frecuencias con las que se trabajan no son tan altas, pueden ser usados a niveles industriales, ya que poseen múltiples ventajas como:

- No requieren adecuación de la señal.
 - Pueden comunicarse directamente con un microcontrolador.
 - Su precio es económico.
 - Son compactos.
- Acelerómetros piezoresistivos:

Son, después de los acelerómetros piezoeléctricos, los más usados a nivel industrial. Su funcionamiento es similar al de los acelerómetros capacitivos; ante una aceleración de entrada se produce un desplazamiento de salida más, en este caso, están constituidos por una o varias galgas extensiométricas, una masa de prueba y unos resortes de soporte. La galga sujeta a la masa sísmica y, al esta recibir una fuerza, produce un desplazamiento proporcional a la fuerza aplicada, lo que deforma a su vez la galga extensiométrica lo cual se traduce como un cambio de resistencia en el sensor. La ventaja de los acelerómetros piezoresistivos es que pueden medir valores de voltaje DC lo que los hace útil en el estudio de impactos, son también usados en el análisis de vibración en el rango de mediana frecuencia.

- Acelerómetros piezoeléctricos:

Según Weber (2012), es el acelerómetro más usado en aplicaciones industriales ya que poseen características como:

- Alto rango dinámico.
- Bajos niveles de ruido.
- Alta linealidad.
- Alto ancho de banda.
- Poco desgaste, ya que no poseen partes móviles.

Su construcción es bastante sencilla, se tiene disco de un cristal piezoeléctrico unido por dos terminales circulares, de forma similar a la de un condensador, y justo encima, tienen una masa de prueba. Al experimentar una fuerza el cristal se deforma lo que produce una diferencia de carga y un voltaje proporcional a la fuerza aplicada. Como la aceleración de la masa de prueba es también proporcional a la fuerza aplicada, la aceleración del acelerómetro será entonces directamente proporcional al voltaje y la carga producida en el cristal.

2.2.4. Procesamiento de señales

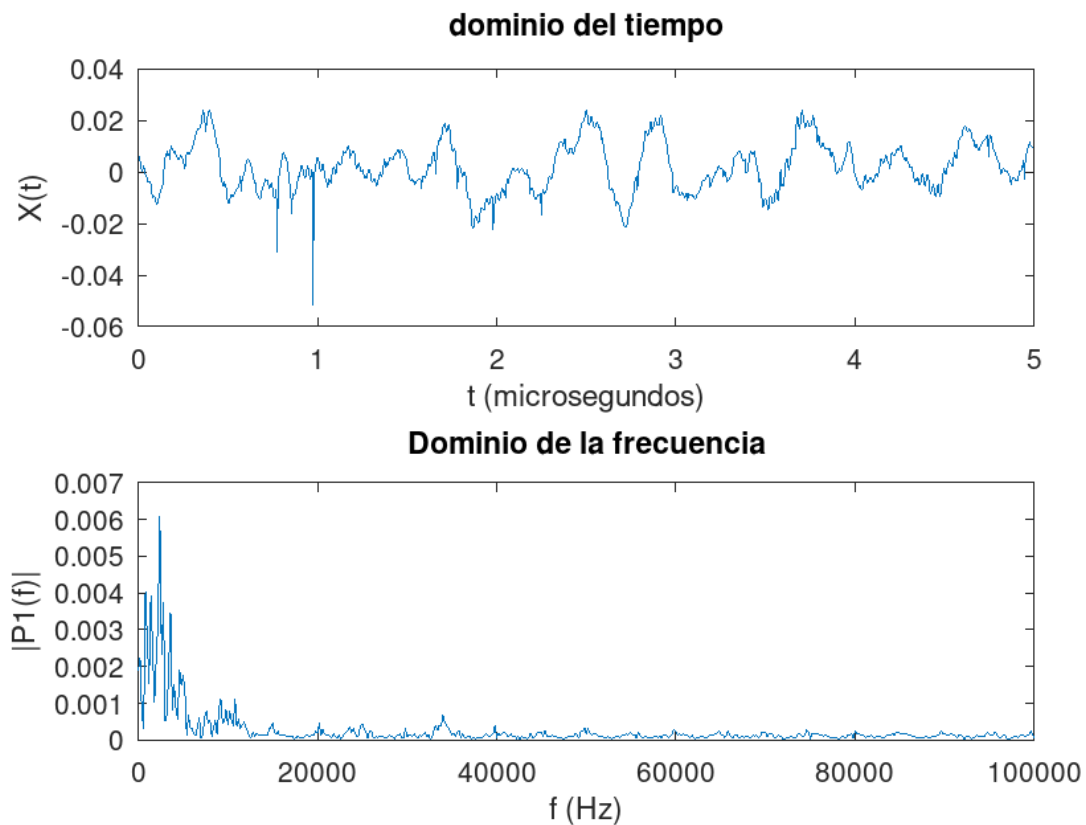
Después de ser almacenada la información, debe ser estudiada, procesada, para lo cual se utiliza una serie de herramientas, técnicas o software especializados a cada necesidad. Este estudio se puede categorizar en dos ramas principales:

- Dominio del tiempo, según Vynith y col. (2019b), es un término utilizado para describir el análisis de funciones matemáticas o señales con respecto al tiempo, la sucesión de estados que atraviesa la señal de forma natural. Los estudios más comunes son en **tiempo continuo** y en **tiempo discreto**.
- Dominio de la frecuencia, según Vynith y col. (2019a), es un término utilizado para describir el análisis de funciones matemáticas, señales o movimientos periódicos respecto a su frecuencia, número de veces que sucede un evento en un periodo. Utilizan transformadas para llevar las funciones o señales del dominio

del tiempo, base, al dominio de la frecuencia, deseado, la más famosa es la **transformada de Fourier**.

Gráficamente se suele entender el **dominio temporal** como la evolución de una señal con respecto al tiempo, es decir su evolución natural, por otro lado, el **dominio frecuencial** muestra los componentes de la señal según la frecuencia en la que oscilan dentro de un rango determinado. En la figura 1 se observan ejemplos gráficos de señales en ambos dominios.

Figura 1: Diagrama de los dominios temporal y frecuencial



Realizada con el Software Octave a partir de los datos de Huang y col. (2018)

El análisis en frecuencia suele ser más utilizado debido a que la mayoría de las fallas poseen frecuencias características y, dado que en el análisis de frecuencia se descompone en frecuencias la señal, se facilita la detección de fallas características, así mismo, la amplitud de la frecuencia es directamente proporcional al nivel de la falla. Por lo tanto, se obtiene un espectro amplio del estado de la pieza. Cabe resaltar que cuando las frecuencias son bajas o muy cercanas entre si, se dificulta determinar

e identificar alguna falla, suele suceder cuando se estudia una falla o evento con frecuencia muy baja o muy cercana a la frecuencia natural de la señal o elemento medido. En estos casos es mejor usar un análisis en el dominio del tiempo que facilita la detección de las fallas.

2.2.5. Herramienta Computacional

Según Wooley JC (2005), una herramienta computacional puede ser definida como cualquier software, sistema de integración, análisis o almacenamiento que ayuda a los científicos o usuarios a solucionar un problema específico en una determinada rama. Pueden variar desde sistemas complejos como compiladores, algoritmos e incluso sistemas operativos hasta herramientas como hojas de cálculos, sistemas de oficina o medios de comunicación. Funcionan mediante la implementación de técnicas y protocolos para solucionar problemas de forma iterativa o con una secuencia de pasos concreta.

Siguiendo este orden de ideas, una gran cantidad de estas herramientas son encontradas en la librería de información más grande del mundo, el Internet. Todas comparten la peculiaridad de que son un **sistema** y, por ende, pueden ser accedidas con facilidad desde cualquier punto con un dispositivo capaz de tener conexión a Internet y un navegador. Esta facilidad se debe a que un **servidor** se encarga de hacer el procesamiento de la información y envía el resultado con un formato específico, típicamente JSON, por “notación de objeto de JavaScript” el cual es un formato de texto sencillo para el intercambio de datos, este se renderiza (proceso para generar una representación gráfica por medio de programas informáticos) en una página Web.

2.2.6. Sistema Web

Los sistemas Web, de acuerdo a AddAppTo (2015) y Themandrak y col. (2018), o también conocidos como aplicaciones Web son sistemas que utilizan la tecnología Web y el Internet o Intranet para transmitir la información y los servicios a usuarios u otros sistemas/aplicaciones. Estos sistemas utilizan los principios del hipertexto para

renderizar la información en cualquier navegador o **página Web** y el poder de los **servidores** para almacenar y procesar la información. Por estas características son independientes de cualquier plataforma o sistema operativo, además de no requerir ningún proceso de instalación, facilitando de esta forma el acceso, la gestión y la rapidez de obtención de información.

Página Web

Una página Web, como se explica en MDN (2021), es un documento accesible desde cualquier navegador con acceso a Internet que puede incluir audio, vídeo, texto y sus diferentes combinaciones. Funciona al usar el protocolo HTTP, conocido usualmente como la Web, y una estructura de hipertexto la cual permite redirigir, enlazar y estructurar el contenido y lo hace fácilmente accesible desde un navegador Web.

Funciona gracias al protocolo HTTP, "Hypertext Transfer Protocol", el cual es la base de cualquier intercambio de datos en la Web y un protocolo de estructura cliente-servidor, esto implica que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web, y es cubierta por el elemento que envía los datos (el servidor). Este protocolo comenzó siendo estático y dirigido usualmente a la transmisión de texto pero se fue convirtiendo en más que eso y, en la actualidad, permite la transferencia de documentos de todo tipo, Scripts, vídeos, entre otros; a tal punto que es fácilmente categorizado como el protocolo más usado en todo el mundo, siendo incluso utilizado como sinónimo de Internet cuando es solo una parte de él.

Debido a la invención de tecnologías, como JavaScript y AJAX, hoy en día es posible tener aplicaciones Web que son programas, junto a una interfaz gráfica, que permiten comunicarse con servidores que realizan la mayor parte del trabajo del desarrollo de aplicaciones complejas que funcionen desde la comodidad de dispositivos móviles. La Web permite, por tanto, facilidad al transmitir información así como el acceso a cualquier contenido desde cualquier dispositivo, en cualquier momento.

La Web suele ser el método de acceso de muchas tecnologías y, si bien en la actualidad

el desarrollo Web usa el mismo estándar de tecnologías, el lado del servidor contempla una variedad mucho más amplia, dado que cualquier aplicación que pueda correr en un ordenador puede ser conectada a una interfaz Web. Teniendo como limitante principal la latencia, tiempo que tarda la información en viajar, una interfaz Web es, para un usuario promedio, una solución cómoda y accesible la cual permite incluso mayor comodidad y facilidad de acceso.

Servidor

Un servidor Web, como lo define PAESSLER (2021), es un ordenador de propósito específico que permite la transición de datos a uno o múltiples clientes Web. Para esto, el dispositivo debe estar configurado para escuchar las solicitudes de los clientes en un entorno red. Esto se logra mediante una aplicación externa o el uso de un sistema operativo dedicado; almacena los archivos necesarios para el procesamiento de información y los datos necesarios para mostrarla, además, se encarga de distribuirla al usuario final.

Los servidores se suelen clasificar según su función y es común que cumplan más de una función, o se encuentren más de un tipo en una red. Algunos de estos son servidores de archivos, impresión, aplicaciones, DNS, **Web**, entre otros. Actualmente, los servidores Web son los más abundantes en el mercado y se caracterizan por alojar la información y los datos de los usuarios a través de Internet o Intranet. Estos responden a las solicitudes de páginas Web u otros servidores basados en esta tecnología.

2.2.7. Base de datos (BBDD)

Una base de datos como dice Microsoft (s.f.) es una herramienta para almacenar y organizar información. Las bases de datos pueden contener información de cualquier tipo y se utiliza para evitar problemas como redundancia y para facilitar el manejo de grandes cantidades de la misma, en fin, una base de datos computarizada es un contenedor de objetos la cual permite:

- Agregar nuevos elementos.
- Editar y mover información en la base de datos.
- Eliminar información.
- Organizarla y visualizarla de diferentes formas.
- Compartirla con otros usuarios, programadores o interesados.

Las bases de datos, según IBM (s.f.-a) se suelen dividir en dos tipos, de acuerdo al tipo de lenguaje utilizado para manipularlo, estas son SQL(“Structured Query Language” utilizado para bases de datos relacionales) y NoSQL (Sus tipos de datos no suelen ser estructurados, aunque algunas bases de datos permiten su estructuración); su utilización o selección depende mucho del tipo de procesamiento que se le dará a la información contenida en ella, algunos otros factores importantes a considerar son la velocidad de escritura-lectura y paralelismo.

2.2.8. MongoDB

MongoDB según IBM (s.f.-c) es un sistema de base de datos no relacional de código abierto el cual utiliza documentos flexibles en lugar de tablas y columnas para almacenar y procesar varios tipos de información. Al ser una solución NoSQL, MongoDB no requiere un sistema de manejo relacional permitiendo esto un modelado mas elástico al momento de guardar la información, además, permite mas libertad en las consultas lo cual, además de simplificar el manejo para los desarrolladores, permite desarrollar un ecosistema fácilmente escalable en aplicaciones y servicios multi plataformas.

Cabe resaltar que el modelo de estructuración de datos que utiliza MongoDB es BSON el cual es formato binario de JSON.

2.2.9. MongoDB Atlas

Como MongoDB (s.f.) especifica en su documentación oficial, MongoDB Atlas es un servicio de base de datos (Database-as-a-Server, DBaaS) el cual permite establecer, llevar a producción y escalar las bases de datos sin preocuparse por el hardware físico, actualizaciones de software y los detalles de configuración.

Es decir, MongoDB Atlas es un sistema completamente manejado en la nube el cual maneja toda la complejidad de llevar a producción el sistema, manejarlo y revisar el estado de sus componentes, además, permite la utilización de cualquier proveedor de servicios (AWS, Azure y GCP).

Cabe resaltar que MongoDB Atlas es un servicio pago, sin embargo, permite la utilización de un "cluster" gratuito, de almacenamiento limitado, para prácticas y aprendizaje.

2.2.10. Interfaz de programación de aplicaciones (API)

Interfaz de programación de aplicaciones es una forma de simplificar el diseño de software al permitir el intercambio de información y funcionalidades de forma rápida y segura. Como explica Education (s.f.) una API permite a las compañías y desarrolladores abrir y expandir las informaciones y funcionalidades que poseen con grupos externos de desarrolladores, compañeros de negocios e incluso departamentos internos dentro de la misma compañía, esto permite separar los desarrollos y trabajar de forma paralela ya que los desarrolladores no necesitan conocer la implementación, simplemente la utilizan como interfaz para comunicarse con otros productos y servicios.

Cabe resaltar que sin estas fuera imposible el desarrollo de muchas aplicaciones populares. Una API funciona al ser un conjunto de normas que definen como se comunicarán las computadoras o aplicaciones entre ellas, es decir, sirve como una capa de abstracción entre el servidor y la aplicación.

2.2.11. Modelo estadístico

Un modelo estadístico, de acuerdo a IBM (s.f.-b), es una representación matemática que permite, mediante ecuaciones, codificar información extraída de los datos y, de esta forma, predecir el comportamiento de un sistema ante situaciones dadas. Funcionan mediante variables aleatorias, una o más variables de las cuales no se tiene completa certeza de su valor o provienen de algún evento aleatorio.

Un modelo estadístico permite inferir ciertas características de un evento, como qué tan probable es tal evento y cómo se distribuyen los valores de la variable. Además, se suele usar como primer paso en generar un modelo más preciso o para la obtención de información cuando no se tiene suficiente, es difícil su acceso o la naturaleza del sistema es extremadamente compleja y dicha tarea es simplemente imposible.

2.2.12. Prueba de bondad de ajuste

Una prueba de bondad de ajuste es una prueba estadística que se realiza a un modelo estadístico con el fin de obtener un valor que permita comparar qué tan bien se ajusta el modelo estadístico seleccionado a los datos experimentales. En general las pruebas de bondad suelen cuantificar las diferencias que existen entre la distribución teórica y la distribución experimental. Según OpenCourseWare (2016) la prueba estándar de bondad de ajuste es la prueba de kolmogorov-Smirnov, siendo esta la más usada porque representa intuitivamente la noción de distancia entre dos distribuciones.

2.2.13. Prueba de Kolmogorov-Smirnov

La prueba de Kolmogorov-Smirnov es una prueba estadística no paramétrica que permite medir la distancia entre dos distribuciones, su principal uso es para calcular pruebas de bondad de ajuste comparando los valores teóricos con los valores experimentales de un modelo estadístico. Para comparar la distancia entre distribuciones primero se consigue la función de distribución acumulada (FDA) para cada una de las distribuciones a comparar, una vez obtenidas las funciones se calcula

el valor absoluto de su diferencia y se consigue el supremo del conjunto (para un número finito de puntos es el mayor valor) y finalmente una vez obtenido este valor se puede calcular con este un valor P donde la hipótesis nula es que las distribuciones provienen de la misma distribución. En el caso de pruebas de bondad solo se posee una muestra y el método sufre algunas modificaciones Corder y col. (2009) explica que la prueba de Kolmogorov-Smirnov de una muestra compara la distribución observada con una distribución empírica modelada con los datos de la muestra.

2.2.14. Lenguaje de programación

Los lenguajes de programación pueden ser definidos, según Aliat Universidades (s.f.) como sistemas estructurados que permiten a las personas o programadores interactuar y dar instrucciones a un programa o software con la finalidad de lograr objetivos.

En la actualidad existe una gran cantidad de lenguajes de programación, algunos desarrollados en la antigüedad que todavía desarrollan un papel importante en los sistemas, como lo son C, C++, Java y otros más modernos como lo pueden ser Go, Rust y Python.

Como se explica en JavaTpoint (s.f.) Los lenguajes de programación suelen ser clasificados en bajo nivel y alto nivel, haciendo referencia a la necesidad de un compilador o intérprete para poder ser ejecutado por la computadora, siendo los de bajo nivel lenguaje de máquina o ensamblador y los de alto nivel los lenguajes comúnmente conocidos, como Python, Java, JavaScript, PHP, C++, Objective C, Cobol, Perl, Pascal, LISP, FORTRAN, Go y Swift. Estos lenguajes de alto nivel se pueden subdividir de acuerdo a la necesidad de un compilador (C, C++, Go, etc.) o de un intérprete (Python, JavaScript, Ruby, etc.) en Escobar (s.f.), se puede leer más de esto. Asimismo, existe otra pequeña subdivisión en el "tipado" del lenguaje, que es la necesidad de especificar el tipo de valor que una variable o constante puede tomar, siendo estas posibilidades un tipado fuerte, medio o débil, también llamados dinámico (débil) o estático (fuerte).

2.2.15. Go

Go es un lenguaje de programación creado en 2007 en Google, específicamente por Robert Griesemer, Rob Pike y Ken Thompson, su sintaxis es similar a la del lenguaje C y tiende a ser dinámico como Python pero con un rendimiento equiparable a los de C o C++. Como se especifica en su página y documentación oficial Golang.org (s.f.): Go es un proyecto de código abierto (open source) y gratuito para hacer a los programadores más productivos, es un lenguaje expresivo, conciso, limpio y eficiente con mecanismos de concurrencia (paralelismo) que permiten fácilmente obtener el máximo rendimiento posible de las máquinas multinúcleo y redes de máquinas actuales, mientras permite una programación flexible y modular, además de un compilado rápido con un recolector de basura (el lenguaje se encarga de liberar memoria no utilizada). Es un lenguaje compilado rápido y con tipado estático (las variables y sus tipos tienen que ser definidos con anterioridad) que se siente como un lenguaje interpretado y con tipado dinámico (variables modificables sin tipo definido, más sencillo de escribir el código pero menos estructurado).

Adicional a esto, este lenguaje tiene un ecosistema de comunidades, librerías y herramientas en constante crecimiento que facilitan el aprendizaje y el desarrollo de código.

2.2.16. Fyne

Fyne es una de las librerías (conjunto de herramientas o paquete) más utilizado en Go para el desarrollo de interfaces gráficas de usuario (GUI), como se indica en su página oficial fyne.io (s.f.) el conjunto de herramientas de Fyne es una herramienta fácil de aprender, gratuita y de código abierto (open source) que permite la creación de aplicaciones gráficas para escritorios, teléfonos y más. Combina el poder y la simplicidad del lenguaje Go con una cuidadosamente creada librería de Widgets (miniaplicaciones o herramientas) que hacen ahora más fácil que nunca la construcción de aplicaciones y su despliegue a producción en todas las plataformas (sistemas operativos, ios, linux, windows, etc.) y tiendas (windows store, Google play, etc.).

2.2.17. HTML

HTML no es considerado un lenguaje de programación propiamente dicho por su incapacidad de realizar acciones de lógica o aritmética, más específicamente y de acuerdo a Mozilla y col. (s.f.-c), es un lenguaje de enmaquetado (una forma de escritura que utiliza elementos sintácticos para dar forma y estructura a la información escrita posteriormente a un proceso de compilación o interpretación por la máquina, otros ejemplos son Latex y Markdown), específicamente un lenguaje de enmaquetado de hipertexto y es el bloque más básico para la Web ya que define y estructura el contenido de la misma.

“Hipertexto” se refiere a los enlaces que se utilizan para conectar la página Web con otras partes, ya sean de la misma página o paginas externas a la misma. Estos enlaces son fundamentales en la Web ya que permiten la actualización y el entrelazamiento de las páginas creadas por otras personas, esto permite una participación activa el la “World Wide Web”.

HTML usa “etiquetas” para dar información adicional el texto, las imágenes, el contenido y permitir una correcta muestra del contenido, estos elementos sintaxicos son por ejemplo: `<head>`, `<title>`, `<body>`, `<header>`, `<footer>`, `<article>`, `<section>`, `<p>`, `<div>`, ``, ``, `<aside>`, `<audio>`, `<canvas>`, `<datalist>`, `<details>`, `<embed>`, `<nav>`, `<output>`, `<progress>`, `<video>`, ``, ``, ``, entre otros.

De esta forma, el mismo texto entre etiquetas distintas, va a tener características visuales distintas, por ejemplo, `<h1>Algo</h1>` es un título, negritas y tamaño de fuente muy superior a `<p>Algo</p>` que sería un párrafo.

2.2.18. CSS

CSS al igual que HTML no es considerado un lenguaje de programación, por las mismas razones, en cambio, como se describe en Mozilla y col. (s.f.-b) es una hoja de estilos en cascada, un lenguaje utilizado para describir y dar estilo a documentos

escritos en HTML o XML, esto lo consigue al describir cómo los elementos deberían ser mostrados en la pantalla, papel, habla u otra tipo de multimedia.

Dado estas características CSS es una parte fundamental en el desarrollo Web, ya sea en su versión para o con la inclusión de Frameworks o librerías como lo son Bootstrap o TailWind CSS.

Funciona mediante la especificación y modificación de las características en entornos, etiquetas o elementos únicos, mediante identificadores, de un entorno HTML o XML y por esto permite la modificación de los atributos. Usualmente, en términos Web, se modifican tamaños de fuente, peso de la misma, color de fondo o de fuente, márgenes, espaciado interno (padding), centrado, entre otras cosas, y mediante la fijación de estas características a atributos particulares se puede desarrollar un entorno gráfico completo, animaciones y transiciones.

2.2.19. JavaScript

Como se describe en Mozilla y col. (s.f.-a), este lenguaje de programación es liviano e interpretado, débilmente tipado y es comúnmente conocido como el lenguaje estándar de los Scripts en las páginas Webs; sin embargo, también puede ser utilizado en otros ambientes como a nivel de servidor y en aplicaciones multiplataforma. JavaScript es un lenguaje basado en prototipos, multiparadigma, dinámico y de ejecución en un solo hilo (no aprovecha la paralelización de los múltiples núcleos) y soporta todo tipo de estilo de programación.

Sus estándares han ido evolucionando con el tiempo y son conocidos como “ECMAScript Language Specification”; comúnmente JavaScript corre del lado del cliente y es utilizado para diseñar como las páginas Web se ven o se comportan ante algún evento específico.

Al ser un lenguaje tan popular y común, posee una gran comunidad y una cantidad muy significativa de Frameworks y librerías, facilitando su aprendizaje y maximizando la cantidad de cosas que permite hacer. Entre las más conocidas están: Node.js para el servidor, React, Angular y Vue para el cliente, además existen librerías como JQuery

que facilitan el lenguaje.

2.2.20. React

React es una librería de JavaScript de código abierto diseñada por Facebook y lanzada por primera vez en 2013, como dice su documentación oficial Facebook Open Source (s.f.) está diseñada para la construcción de interfaces de usuario, tiene una filosofía de página única (singlepage) pero puede ser utilizada para desarrollar aplicaciones de múltiples páginas. Esta librería tiene la característica de ser declarativa y basada en componentes.

Como se mencionó anteriormente, React hace sencilla la creación de interfaces de usuario al permitir la combinación de JavaScript con XML-HTML en "jsx" dando la posibilidad de devolver y renderizar HTML desde funciones con componentes lógicos. Además, es completamente modular por lo que se pueden desarrollar individualmente los elementos de la interfaz y juntarlos con total facilidad, estos elementos son llamados componentes y utilizan todas las propiedades de JavaScript y la manipulación del DOM (Document Object Model) para mostrar y actualizar la información cuando es necesario.

2.2.21. Python

En su página oficial, específicamente en sus preguntas frecuentes P. S. Foundation (s.f.) se especifica que Python es un lenguaje de programación interpretado, interactivo y orientado a objetos, este soporta muchos paradigmas además del orientado a objetos, como lo son el procedimental y el funcional. Python incorpora funcionalidades como módulos, excepciones, un tipado dinámico y un muy alto nivel de dinamismo en los tipos de datos y clases, asimismo, combina un poder computacional bastante alto con una muy clara sintaxis, librerías, sistemas y compatibilidad con todos los sistemas operativos, además de una facilidad para extenderse a los lenguajes C o C++.

Cabe resaltar que Python es completamente gratuito y tiene una gran capacidad, dada su gran cantidad de librerías y Frameworks, para cumplir muchas tareas, como

lo son estadística, sistemas Webs, ciencia de datos entre otros.

2.2.22. Django

Django es un Framework Web de alto nivel de Python el cual fomenta un desarrollo rápido, limpio y pragmático. Como se especifica en su documentación oficial, D. S. Foundation y col. (s.f.) Fue creado para desarrolladores con experiencia que necesitan un desarrollo rápido y completo en un proyecto Web, con la finalidad de permitir al programador desarrollar la aplicación propiamente dicha, es completamente gratuito y de código libre.

Entre las características de Django son su gran velocidad para el diseño, gran cantidad de extras y tareas adicionales fácilmente configurables (autenticación, administración, etc.), seguridad (previene errores comunes como SQL injection y cross-site scripting-requests), versatilidad y facilidad de escalabilidad.

Cabe destacar que Django permite la inclusión de librerías de Python para cualquier ámbito y existen algunas específicamente diseñadas para facilitar el desarrollo con este Framework, por ejemplo, Django rest Framework, para la creación de APIs desde el servidor de Django y Django-cors-headers que permite modificar y controlar el acceso de solicitudes a API desde clientes.

2.2.23. Numpy

Numpy es un paquete-extensión de Python diseñado para la computación científica. Como se explica en su documentación oficial NumPy (s.f.) es una herramienta de computación numérica que ofrece una gran cantidad de funciones matemáticas, generación de números aleatorios, rutinas algebraicas de álgebra lineal, transformadas de Fourier, el tratado de arreglos N-dimensionales, vectorización e indexación de los mismos, todo esto a un gran rendimiento y una facilidad notable de su uso . Además de todo esto, esta es una herramienta gratuita de código abierto.

2.2.24. Pandas

Pandas es otra herramienta dedicada al cálculo numérico, es una herramienta rápida, poderosa, flexible y de fácil utilización, además de código abierto, utilizada para el análisis y la manipulación de datos, es creada en Python. Entre sus características destacan su velocidad y eficiencia con objetos de tipo DataFrame para la manipulación de la información, sus herramientas para la lectura y escritura de información en distintos formatos, entre los cuales destacan los formatos CSV, Excel, SQL DataBase y el HDF5, facilidad para la mutabilidad, alineamiento, filtración y eliminación, inserción y separación de datos, y una muy alta optimización con puntos de código críticos escritos en Cython o C. Además de esto, otras características adicionales son expuestas en su documentación oficial NumFOCUS (s.f.)

2.2.25. SciPy

SciPy es una librería-colección de algoritmos para la computación científica en Python, como se explica en su sitio oficial SciPy (s.f.), está desarrollado de forma abierta en GitHub a través del consenso y el trabajo de la comunidad científica de Python y de la comunidad de SciPy. Esta herramienta provee estructuras de datos y algoritmos para optimización, integración, interpolación, problemas algebraicos, ecuaciones, ecuaciones diferenciales, estadística, entre otros. Está escrita en lenguajes de medio-bajo nivel, como lo son Fortran, C y C++, haciendo sus implementaciones bastante optimizadas y permitiendo el uso de código compilado con la flexibilidad y facilidad de uso de Python.

2.2.26. FastApi

Ramírez (s.f.) define FastApi como un Framework moderno y rápido, con alto rendimiento, diseñado para la construcción de API Web en Python, tiene una versión mínima de 3.6. Sus características principales son:

- Velocidad, equiparable con Node.js y Go haciéndolo uno de los Frameworks de

Python mas rápidos actualmente.

- Velocidad de escritura, permitiendo alcanzar un aumento en el desarrollo de hasta un 300 %.
- Facilidad, está desarrollado para ser intuitivo, fácil, permitiendo reutilización y a su vez eliminando hasta un 40 % de los errores ("bugs") inducidos por humanos.
- Utiliza estándares de código abierto como lo son "OpenApi y "JSON Schema.

2.2.27. Octave

Octave es un software originalmente escrito por John W.Eaton y muchos otros, esto es debido a que es un lenguaje gratuito y constantemente incorpora funciones o correcciones hechas por la comunidad. Como se especifica en su documentación oficial Eaton. (s.f.), es un lenguaje de programación de alto nivel, orientado primordialmente a la realización de cálculos numéricos. Permite el uso de una interfaz o la terminal para resolver ecuaciones lineales, no lineales, problemas numéricos además de conversiones y transformadas matemáticas. Es completamente compatible con el lenguaje Matlab y además puede ser utilizado como un lenguaje orientado a procesos.

Cabe resaltar que Octave posee una gran gama de librerías o módulos escritos en C++, C, Fortran entre otros lenguajes, y es completamente gratuito y redistribuible.

2.2.28. Git

Es un software de control de versiones diseñado por Linus Torvalds pensado para la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones especialmente útil cuando estas tienen un gran número de versiones y archivos. Como explica community (s.f.), Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para manejar eficientemente tanto pequeños como grandes proyectos de forma rápida y eficiente. Se diferencia de otros sistemas por características como ramificaciones locales baratas, en términos computacionales

espacio y velocidad, múltiples flujos de trabajo en áreas de ensamblaje convenientes (Permite trabajar tanto en clientes gráficos como en la terminal).

2.2.29. GitHub

GitHub es una plataforma para el montaje de código y el control de versiones basada en Git. Se utiliza para la creación de código fuentes ya que permite y facilita la interconexión entre los programadores, como se explica en Inc (s.f.) GitHub permite incrementar la velocidad del desarrollador, además de asegurar cada paso, automatizar los espacios y flujos de trabajo, facilitando de esta forma los patrones de desarrollo e integración continua y permitiendo la creación de equipos de trabajo sin el impedimento de la ubicación geográfica.

2.2.30. Hosting

Según sistema TDC (s.f.) un Hosting es el servicio que permite que un sitio Web o dominio permanezca en internet, facilitando el guardar la información de un sitio y acceder a este, además en este espacio se puede acceder a API o servidores.

Existen múltiples tipos de Hosting y múltiples servicios o empresas que se dedican a prestar este servicio, algunas con características completamente pagas y otras que permiten la utilización del sitio, con recursos mas restringidos de formas gratuitas y la ampliación de estos e inclusión de funcionalidades extras mediante la contratación de un plan con determinados recursos y políticas.

Entre las empresas mas conocidas se encuentra DigitalOcean la cual, como se explica en su sitio oficial DigitalOcean (s.f.) se dedica a alquilar servicios de cómputo en la nube los cuales son robustos y escalables, además de contener una muy buena documentación. DigitalOcean permite el agregar API además de servidores mas complejos, los últimos mediante una implementación de una maquina virtual con "ubuntu server" como sistema, y facilitar de esta forma el acceso a la información entre servicios, microservicios o servidores.

CAPÍTULO 3

MARCO METODOLÓGICO

Como fue explicado en el primer capítulo, se pretende hacer una herramienta computacional para el análisis de la vibración en motores eléctricos alimentada mediante datos de una simulación digital. Partiendo de esto, se comenzará con la definición de los siguientes aspectos.

3.1. NATURALEZA DE LA INVESTIGACIÓN

El presente trabajo es clasificado como Proyecto Especial, puesto que según Hernández (1998) lleva a:

Trabajos que lleven a creaciones tangibles, susceptibles de ser utilizadas como soluciones a problemas demostrados, o que respondan a necesidades e intereses de tipo cultural. Se incluyen en esta categoría los trabajos de elaboración de libros de texto y de materiales de apoyo educativo, el desarrollo de software, prototipos y de productos tecnológicos en general, así como también los de creación literaria y artística.

3.2. TIPO DE INVESTIGACIÓN

De acuerdo con la clasificación el tipo de investigación de este trabajo se cataloga como investigación aplicada, puesto que "persigue fines inmediatos y concretos a través de la búsqueda de un nuevo conocimiento técnico con aplicación inmediata a un problema determinado" Vélez (2005).

3.3. FASES DE LA INVESTIGACIÓN

En función de los objetivos específicos, se pueden definir las fases que conforman la investigación:

3.3.1. Definición de los requerimientos de usuario

Esta fase tiene relación directa con la concepción del proyecto ya que fue el planteamiento de los problemas y posibles soluciones, separación de vistas, mecanismos de análisis, entre otros. Sin una concepción de las estructuras de datos y los lenguajes necesarios para elaborarla. Como en todo proyecto, esta fase requirió de la interacción con el cliente (ingeniero mecánico que propuso la idea) con la finalidad de puntualizar los requerimientos mínimos necesarios para obtener una aplicación viable.

De esta manera, la primera fase ha constado de las siguientes actividades:

- Selección de la cantidad de vistas óptimas para facilitar el estudio y el monitoreo de la planta, siendo esto decidido a 3, una vista **general** del estado de los motores en la planta, una **específica** en donde se observa el comportamiento de un motor, una **exhaustiva** que añade información a la específica, gráfica en el dominio de la frecuencia, agregada para mejorar la experiencia de usuario, por la demora necesaria para generar este estudio.
- Definición de la estructura mínima necesaria para poder evaluar un motor, en términos de evolución histórica de la vibración, con base en una tabla histórica o gráfica (histórica).
- Definición del tipo de transformada de Fourier a graficar para el análisis exhaustivo (por gustos y costumbres en la industria).
- Entrega de una base de datos histórica con mediciones reales y esporádicas del estado de los motores en una fábrica.

- Especificación de los parámetros a considerar para determinar un nivel básico del estado de un motor, en términos de velocidad y aceleración. Se debe resaltar que estos valores difieren de los establecidos a nivel teórico y recomendado en los estándares por factores inherentes a las empresas (malas instalaciones, utilización de bases artesanales, políticas de daños mínimos) los cuales amplían los rangos teóricos.
- Selección de las unidades en las que se tomarán las medidas, siguiendo estándares, y siendo $\frac{mm}{s}(rms)$ para la velocidad y g para la aceleración.

Es importante recalcar el hecho de que los parámetros utilizados, tanto para el establecimiento de niveles de daño en el modelo estadístico como en la vista general para distinguir el estado de los motores, son el resultado empírico asociados a años de estudios del comportamiento de los motores en la planta utilizada. Esto se tomó en consideración al momento del diseño y se puede modificar, vía modificación de constantes en el código, los rangos de valores en velocidad y vibración adecuados a los estándares de cada planta o ingeniero que utilice el sistema.

3.3.2. Revisión documental

Relacionada parcialmente con los primeros dos objetivos de la investigación, se basó en los estudios necesarios para poder tomar la decisión del tipo de modelo y lenguajes a utilizar, el planteamiento de requerimientos para los lenguajes, dada la gran variedad de posibilidades disponibles para las distintas tareas a realizar; de esta forma, se pueden enumerar las siguientes actividades realizadas:

- Recolección de información de diversos lenguajes de programación.
- Estudio de estadística y los distintos modelos y distribuciones existentes.
- Estructurar los criterios y requerimientos mínimos para poder seleccionar las herramientas a utilizar.

3.3.3. Análisis de la información

Fase previa a la implementación pero relacionada al primer objetivo de la investigación. Se procedió a analizar la información en las bases de datos, comparar con los requerimientos mínimos por el cliente y estructurar mas detalladamente la nueva base de datos (creada posteriormente en MongoDB), los parámetros necesarios para el modelo estadístico y la selección del tipo, orientado a una distribución de probabilidad. Asimismo, se tomaron decisiones conforme a los lenguajes utilizados y la estructura de microservicios que se utiliza en el desarrollo. De esta forma, se distinguen los siguientes puntos en esta fase:

- Organización de la información recolectada.
- Toma de decisión de los lenguajes y herramientas a utilizar, basada en los criterios previamente establecidos.
- Estructuración de los modelos para las bases de datos.
- Justificación de las decisiones tomadas.
- Estructuración y separación de las tareas a realizar por los distintos microservicios (sensorica, Web, BBDD).

3.3.4. Diseño del modelo estadístico

Corresponde al segundo objetivo del trabajo pero guarda relación con los objetivos 3 y 4. Se utilizaron técnicas de exploración de data para la selección de las variables de interés así como la formulación de hipótesis y se usaron pruebas estadísticas para seleccionar las distribuciones de probabilidad que mejor modelan las variables de interés. Se realizaron las siguientes tareas:

- Limpieza de los datos.
- Análisis exploratorio de la data.

- Cálculo de estadísticos de interés.
- Cálculo de pruebas de bondad de ajuste para cada variable de interés.
- Selección de las distribuciones que mejor modelan cada variable.

3.3.5. Diseño de las vistas para mostrar la información

Este apartado guarda relación con los objetivos 5 y 6. Se trabajo en la estructuración y diseño gráfico de las vistas, además de las funcionalidades implementadas para poder ser creada la página Web. Las tareas realizadas fueron las siguientes:

- Toma de decisión en utilizar una aproximación multipágina (multipage app).
- Diseño básico, barra de navegación, encabezado y pie de página.
- Diseño de la vista General y su funcionalidad.
- Diseño de la vista Especifica y su funcionalidad.
- Diseño de la vista Exhaustiva y su funcionalidad.

3.3.6. Implementación

Dada la gran cantidad de tareas a desarrollar, la implementación se subdivide en un gran grupo de tareas, con la finalidad de cumplir los objetivos 2,3,4,5,6 estas divisiones, y subdivisiones, pueden ser expresada de la siguiente forma:

- Generación del modelo estadístico:
 1. Inspección manual de la data.
 2. Limpieza de las variables de interés.
 3. Elaboración de histogramas para observar la distribución de los datos.
 4. Seleccionadas las variables velocidad horizontal, velocidad vertical y aceleración como variables para el modelo.

5. Cálculo de correlación entre variables.
 6. Cambio de variables para reducir la correlación entre velocidad horizontal y velocidad vertical.
 7. Búsqueda de distribuciones que mejor modelan la data, utilizando múltiples pruebas de Kolmogorov-Smirnov.
 8. Selección de distribuciones de Burr tipo III para modelar la magnitud de la velocidad y aceleración y distribución normal para modelar el ángulo.
 9. Cálculo de parámetros de las diferentes distribuciones. Creación de API en FastAPI para servir los datos.
 10. Conexión de la API con el microservicio de Go.
- Elaboración de la base de datos:
 1. Registro y creación del cluster en MongoDB Atlas.
 2. Creación de la base de datos "tesis" y las colecciones "MotorData" y "MotoresInDB".
 3. Llenado mediante el servidor de sensorica.
 - Microservicios de sensorica:
 1. Creación de las estructuras de datos necesarias para manejar la comunicación con MongoDB.
 2. Obtención de un certificado y TLS/SSL.
 3. Creación del servidor Https.
 4. Creación de un cliente de sensorica, para facilitar la emulación de los sensores
 5. Implementación de las conexiones bidireccionales para el tráfico de información.
 6. Creación del punto de conexión "/sensormessage" para permitir la conexión de múltiples clientes y recibir la información.
 7. Creación del punto de conexión "/exhaustive?idMotoridentificador" para poder enviar la información de la vista exhaustiva.

8. Intercomunicaciones internas en el servidor para poder comunicar, solicitar y verificar información entre los dos puntos de conexión.
 9. Verificaciones de seguridad y conexiones.
- Análisis en frecuencia:
 1. Obtención de las mediciones del servidor de Sensorica.
 2. Aplicación del algoritmo FFT para obtener la nueva información.
 3. Graficar lo obtenido en le punto anterior.
 - Servidor Web:
 1. Configuración para el envío de información estática (HTML, CSS, Js, imágenes).
 2. Creación del punto de conexión “/” para la vista general.
 3. Creación del punto de conexión “/especifica?IdMotoridentificador” para la vista específica.
 4. Creación del punto de conexión “/exhaustiva?IdMotoridentificador” para la vista exhaustiva.
 5. Creación de gráficas históricas.
 6. Solicitud de información y análisis en frecuencia, creación de la gráfica de frecuencia.
 7. Establecimientos de los puntos de conexión para las API que suministran las informaciones necesarias para las vistas (uno para cada vista).
 - Muestreo de información, mediante las vistas:
 1. Solicitudes de información a las API (fetch).
 2. Configuración de los valores y los niveles de daño.
 3. Establecimiento del motor svg con colores dependiendo del nivel de daño.
 4. Mecanismo de paginación y búsqueda para hacer mas fácil y agradable la interfaz de usuario.
 5. Creación de la vista general.

6. Implementación de la tabla y la funcionalidad para exportar a Excel.
 7. Solicitud y establecimiento de las imágenes-gráficas.
 8. Implementación de la vista específica.
 9. Implementación de la vista exhaustiva.
 10. Diseño y estilizado mediante CSS.
- Interconexiones:
 1. Interconexiones entre cliente y servidor de sensorica, mediante Http2 para poder establecer una comunicación bidireccional (full dúplex).
 2. Interconexión entre servidor de sensorica y API de modelo estadístico para información normal, grupo de 10 mediciones de velocidad y aceleración por cada sensor.
 3. Interconexión entre servidor de sensorica y API del modelo estadístico para información exhaustiva, grupo de 1K de mediciones de aceleraciones, por sensor registrado al motor.
 4. Interconexiones entre el servidor de sensorica y la base de datos (MongoDB Atlas).
 5. Interconexiones entre el servidor Web y la base de datos (MongoDB Atlas).
 6. Interconexión entre servidor Web y servidor de sensorica, para solicitar la vista información asociada a la vista exhaustiva.
 7. Interconexión entre cliente Web y servidor Web para obtener información de archivos estáticos.
 8. Interconexión entre cliente Web y API del servidor Web para obtener la información referente a las vistas general, específica y exhaustiva. Una interconexión y una API para cada vista.

3.3.7. Comprobar los resultados

Esta fase hace referencia al último objetivo de la investigación: Comprobar los resultados de la herramienta de análisis. Dada la extensión de la herramienta, estas

pruebas se pueden desglosar en dos niveles, resultados finales (evaluación del análisis de la herramienta) y pruebas técnicas (también llamado testing). Esto da origen a las siguientes tareas:

- Testing: La elaboración modular del sistema permite corroborar su funcionamiento de múltiples formas, dado que unas pruebas automatizadas escapan de esta investigación (por factores de tiempo y cantidad de pruebas y complejidad de las mismas a nivel de código) se toman pruebas manuales de sistemas e interconexión entre los mismos:
 1. Prueba de no unicidad de la información, al haber múltiples clientes de sensorica enviando información sobre el mismo motor.
 2. Prueba de falla en la conexión de la base de datos.
 3. Prueba de solicitud de vista exhaustiva a un motor sin conexión establecida.
 4. Prueba de petición no completada en el cliente Web, información inválida (motor no existente) o error del servidor.
- Evaluación de resultados:
 1. Corroboración del análisis en nivel de daño básico (vista general) que se ajuste a los parámetros requeridos por el cliente.
 2. Corroboración de las gráficas históricas.
 3. Corroboración de la gráfica de la transformada de Fourier.

3.4. METODOLOGÍA

3.4.1. Metodología de desarrollo de software

Para la implementación de los distintos sistemas que componen la herramienta computacional propuesta, se necesitar seguir una metodología de diseño de software. Como la herramienta se clasifica como una aplicación Web, las metodologías utilizadas en el desarrollo de este tipo aplicaciones son aplicables a este trabajo.

Las aplicaciones Web cuentan con muchas similitudes a las de aplicaciones de computadores personales, sin embargo, una de sus principales diferencias es que estas están en un estado de constante cambio por lo tanto se pierde la noción de versiones, los cambios toman efectos de forma inmediata y de forma gradual. Esta diferencia es producto de que su función principal es la de transmitir información. Como esta información cambia de forma constante los requisitos de la aplicación suelen también ser variables. Todo esto unido al hecho de que los cambios en la aplicación pueden realizarse con la menor fricción posible, los cambios se pueden observar al volver a cargar la página, hace que este tipo de aplicación exista en un estado de evolución continua. Según Pressman (2002) la evolución constante de las aplicaciones Web puede ser comparada con la jardinería, se hace un trabajo inicial el cual seria equivalente a sembrar un jardín y una vez se tenga el sitio implementado, se debe realizar el trabajo de mantenerlo que seria equivalente a regar y abonar las plantas.

Debido a lo presentado anteriormente, para la elaboración de los diferentes componentes de la herramienta se realizó un desarrollo de forma continua, en donde la aplicación tendrá la capacidad de evolucionar si algún día cambian los requisitos de la misma. Para esto se utilizó la ayuda de **Git** como sistema de control de versiones, lo cual permitiría integrar los diferentes cambios de forma mas eficaz y facilitar la cooperación durante el desarrollo de la aplicación. Para la elaboración de los diferentes componentes se usa también un proceso iterativo, cada vez que se implemente un componente se comprueba el funcionamiento del mismo y se corrige, de ser necesario.

Este modelo de implementación posee una estructura muy marcada, la cual se adapta a las metas, usuarios finales y a la filosofía de navegación que se elija. Según Pressman (2002) esta se puede desglosar como:

- Diseño arquitectónico, el cual consiste en la definición de la estructura global, las plantillas y parte del patrón de diseño que se utilice para estructurar la red. Específicamente, se trata de cimentar las bases para facilitar la creación del contenido y se estructura la forma de navegación por el sistema, además de los componentes que la integren.
- Diseño de navegación, en esta fase se eligen las rutas de navegación que permiten

el acceso al contenido, además, se hacen las distinciones con respecto a los usuarios que accedan a la red y, por ende, las funcionalidades y/o permisos a los que tendrán acceso.

- Diseño de la interfaz, se refiere al diseño gráfico, estético y a las facilidades que se le dan a los usuarios, dado que esta es la primera impresión que se da, puede significar la diferencia entre el uso o no de la aplicación.
- Cabe destacar que, en la actualidad, se consideran mas fases que las mencionadas por Pressman, las dos mas resaltantes son la **Fase de programación** y la **Fase de Testeo**, aunque la última es nombrada como un proceso posterior al desarrollo. Estas dos fases son consideradas en la actualidad dados los constantes requerimientos de implementaciones adicionales o modificaciones a las ya existentes. Para permitir la integración de la mismas sin exponerse al colapso del sistema, se suelen utilizar **test** automatizados los cuales deben ser diseñados e implementados.

3.5. RECURSOS

- Computadoras portátiles.
- Documentación.
- Base de datos de la vibración en motores eléctricos.

CAPÍTULO 4

ANÁLISIS Y DISCUSIÓN DE RESULTADOS

4.1. PROCEDIMIENTO DE RECOLECCIÓN DE INFORMACIÓN

Esta parte del proyecto es bastante extensa así que debe de ser dividida en dos categorías, investigación de campo e investigación documental.

La investigación de campo, según Arias (2006) es aquella que consiste en la recolección de datos directamente de los sujetos investigados, o de la realidad donde ocurren los hechos, sin manipular o controlar variable alguna, es decir, el investigador obtiene la información pero no altera las condiciones existentes. De allí su carácter de investigación no experimental.

De esta forma, análisis de campo juega un rol importantísimo para obtener la información y los requerimientos mínimos para el diseño de la interfaz de usuario y del modelo estadísticos.

Asimismo, la investigación documental, según Arias (2006) es un proceso basado en la búsqueda, recuperación, análisis, crítica e interpretación de datos secundarios, es decir, los obtenidos y registrados por otros investigadores en fuentes documentales impresas, audiovisuales o electrónicas, siendo esta fundamental para la creación de clasificaciones, comparaciones, tablas y la implementación propiamente dicha del trabajo. Para esto se revisaron libros, trabajos de investigación, documentos de fuente electrónicas, cursos, entre otros.

4.2. ELECCIÓN DE LAS HERRAMIENTAS Y LENGUAJES

Se debe resaltar que los valores otorgados a las características en los lenguajes fueron decisión tomadas a juicios y criterios personales entre los miembros del trabajo y orientada a conocimientos obtenidos en los momentos de investigación, y pre-existentes.

Dada la gran cantidad de actividades y los requerimientos que tenían, se hizo un estudio de los lenguajes y herramientas mas utilizados en la actualidad para tareas similares y junto a esto, se establecieron ciertos criterios y requerimientos mínimos para cada tarea a realizar, de esta forma, se tomaron las siguientes decisiones:

4.2.1. Servidor

La decisión del servidor se puede dividir de acuerdo a los microservicios a realizar, es decir, el servidor dedicado a sensorica y el servidor Web. En ambos casos se manejaron los mismos lenguajes, asimismo, debido a las características variantes entre estos y la posibilidad, dada la arquitectura distribuida empleada de utilizar múltiples herramientas, se optó por utilizar el que mejor cumplía los requerimientos establecidos; como se observa en la tabla 1, se comparan los lenguajes Golang, Python con Framework Django y NodeJs, esto es debido a que los servidores siguen un sistema Web y es mas fácil el desarrollo en estos lenguajes especializados que en sistemas como Apache o Nginx mas robustos pero generales.

Para el microservicio de sensorica, las características mas importantes fueron el paralelismo y la facilidad de implementar una conexión Http2, por la necesidad de una conexión full dúplex; por estos motivos Golang fue un claro ganador dado que es un lenguaje diseñado para microservicios y paralelismo y el uso de librerías facilitan increíblemente el establecer una conexión Http2.

Por otro lado, en términos del servidor Web, las necesidades giraban en torno a la facilidad de implementación, robustez y escalabilidad que otorgaban los lenguajes

Tabla 1: comparación entre los posibles lenguajes utilizables para servidores

Distintas opciones manejadas al momento de desarrollar los servidores.

Característica	Golang	Django	NodeJs
Velocidad	10	7	8
Facilidad de implementación	8	9	7
Robustez	6	9	7
Escalabilidad	10	10	9
Paralelismo	10	-	2
Facilidades a conexión Http2	9	7	7

siendo en este caso Python + Django el ganador indiscutible.

Cabe resaltar que paralelismo hace referencia al mejor uso posible de los recursos de hardware del servidor, por la gran cantidad de conexiones continuas y tareas adicionales que deben ser realizadas y estar establecidas, no a la cantidad de peticiones que recibe el servidor.

4.2.2. Cliente

En términos de clientes también se debe hacer diferencia entre los 2 sistemas creados, cliente de sensorica y cliente Web. El cliente de sensorica se implementó en Golang por las mismas razones dadas para el servidor de sensorica.

Por otro lado, en la actualidad los clientes Web están compuestos casi en su totalidad de una combinación de HTML-CSS-JavaScript, existen casos en los que no se usa JavaScript o clientes escritos en otros lenguajes e interpretados en la web; de esta forma, la decisión está en qué Framework de JavaScript se va a utilizar, existe una gran variedad que cubre desde JavaScript "vainilla", sin Framework y JQuery, que otorgan interactividad pero se complica rápidamente cuando la aplicación crece, hasta los 3 gigantes en la actualidad, ReactJs, Angular y VueJs, que facilitan y agilizan increíblemente el proceso de desarrollo. Como se explica en la tabla 2, La razón mas importante en la toma de la decisión fueron conocimientos previos con ReactJs.

Tabla 2: comparación entre los posibles lenguajes utilizables para clientes Webs

Distintas opciones manejadas al momento de desarrollar los clientes Web.

Característica	JavaScript	React	Otros JsFrameworks
Facilidad de implementación	1	7	7
Conocimientos previos	6	5	-

4.2.3. Modelo estadístico

La elección de herramientas para el modelo estadístico se realizó en dos etapas, primero fue necesario escoger qué herramientas iban a ser usadas para el análisis exploratorio y después fue necesario escoger en cuales iba a ser implementado.

Para el análisis de datos hoy en día se cuenta con una amplia gama de herramientas pero por cuestiones de costo y versatilidad el enfoque fue realizado a lenguajes de programación usados en análisis de datos. En específico se compararon Julia, Python y R de acuerdo a su velocidad de implementación, conocimiento previo, versatilidad y madurez del lenguaje, como se puede observar en la tabla 3

Tabla 3: Comparativa de posibles lenguajes para el análisis exploratorio

Característica	Python	R	Julia
velocidad de implementación	8	9	5
Conocimiento previo	9	7	2
Versatilidad	9	3	7
Madurez del lenguaje	10	8	4

Python fue un claro ganador, R era una buena segunda opción pero su mayor limitante es el hecho que es un lenguaje de casi uso exclusivo para estadística y la versatilidad de Python permite extender el modelo en un futuro.

Finalmente para la implementación del modelo se tenían dos opciones, hacerlo directamente en Go o utilizar también Python y se optó por la segunda principalmente porque la mayoría de los modelos ya están implementados en Scipy mientras que Go no dispone de muchas librerías enfocadas en la estadística y el código se tenía que escribir de forma manual. Para interconectar estos dos componentes se decidió utilizar

una API Web con FastAPI porque es por mucho la forma más rápida de implementar una API Web en Python.

4.2.4. Análisis en frecuencia

En el caso del análisis de frecuencia se requiere una herramienta que permita tanto el cálculo de la transformada rápida de Fourier como su visualización, siendo el segundo la mayor limitante a la hora de escoger las herramientas porque la mayoría de los lenguajes poseen alguna librería para el cálculo de la transformada de Fourier pero el software de visualización es más escaso. Surgieron tres opciones naturales Python, Octave y Julia los cuales fueron comparados de acuerdo a su facilidad de conexión con el resto de componentes, modificabilidad de las gráficas, velocidad de ejecución y madurez del lenguaje. Esta comparativa se puede ver en la tabla 4

Tabla 4: Comparativa de posibles lenguajes para el análisis de frecuencia

Característica	Python	Octave	Julia
Facilidad de conexión	10	2	6
Modificabilidad gráficas	9	5	7
Velocidad ejecución	7	6	9
Madurez del lenguaje	10	7	4

En el caso del análisis de frecuencia se requiere una herramienta que permita tanto el cálculo de la transformada rápida de Fourier como su visualización, siendo el segundo la mayor limitante a la hora de escoger las herramientas porque la mayoría de los lenguajes poseen alguna librería para el cálculo de la transformada de Fourier pero el software de visualización es más escaso. Surgieron tres opciones naturales Python, Octave y Julia los cuales fueron comparados de acuerdo a su facilidad de conexión con el resto de componentes, modificabilidad de las gráficas, velocidad de ejecución y madurez del lenguaje.

4.3. IMPLEMENTACIÓN DEL MODELO ESTADÍSTICO

El diseño del modelo comenzó con la limpieza de los datos y la selección de las variables de interés de la base de datos proporcionada. Las variables de interés fueron la velocidad horizontal, la velocidad vertical, la aceleración, la potencia del equipo, el tipo de equipo y el código del mismo. Una vez obtenida la data se le realizó un análisis exploratorio, el cual consistió principalmente en la elaboración de gráficas y sacar estadísticos de las variables de interés de lo cual se obtuvieron las observaciones que se incluyen a continuación.

Las variables de velocidad horizontal, velocidad vertical y aceleración son siempre positivas, todas muestran en su distribución una asimetría con tendencia hacia la izquierda, lo cual las hace similar a distribuciones como la exponencial, la weibull o la log-normal. Cuando se calcula la correlación entre las variables velocidad horizontal y vertical obtenemos una correlación de Pearson del 56,71 %, el cuadrado de este valor o R^2 (también conocido como coeficiente de determinación) es del 32,16 % y se puede interpretar como que el valor de la velocidad horizontal predice 32 % del valor de la velocidad vertical, estos valores son muy elevados para considerar las variables como independientes. En el caso de la aceleración las correlaciones no fueron significativas.

Debido a la correlación considerable entre las dos velocidades no era posible modelar las variables como independientes pero tampoco era tan elevada para realizar una regresión lineal, por lo tanto se recurrió a buscar un cambio de variables en el cual desapareciera la dependencia entre las dos variables aleatorias. Como la velocidad es un vector y se tienen sus componentes, un cambio de variables lógico es el de coordenadas polares ya que como es una transformación biyectiva no se pierde información aplicándola y, a su vez, es fácil de interpretar. La correlación entre las nuevas variables magnitud y ángulo fue del $-25,18\%$ y un valor de R^2 del $6,34\%$ lo cual para efectos prácticos se puede considerar como dos variables aleatorias independientes. Similarmente con las velocidades con componentes cartesianos la correlación con la aceleración es no significativa.

Una vez obtenidas las tres variables de interés, para modelar se buscó la distribución

que mejor se aproxima a los datos, la escogencia de la distribución de probabilidad por lo general no es un proceso cuantitativo sino que el investigador conociendo en profundidad el problema escoge entre un grupo de distribuciones conocidas, cuál se adapta mejor al problema planteado, pero debido a que hoy en día existen herramientas computacionales para la estadística este proceso se puede hacer un poco más cuantitativo. Para la selección de los modelos de cada variable se iteró entre 84 distribuciones pertenecientes a Scipy y se aplicó a cada una la prueba de Kolmogorov–Smirnov, se filtraron aquellas distribuciones con valores P mayores a 5 % y, finalmente, se ordenaron de mayor a menor, debido a que la hipótesis nula de la prueba de Kolmogorov–Smirnov es que si las distribuciones poseen la misma distribución un valor elevado de P indica que la distribución, se aproxima al valor teórico.

El modelo estadístico seleccionado fue el siguiente, se tienen 3 variables independientes: magnitud de la velocidad, ángulo de la velocidad y aceleración. Para la magnitud de la velocidad se seleccionó una distribución Burr de tipo III debido a que es una distribución netamente positiva y la prueba de Kolmogorov dio un valor P del 78,01 % siendo este el segundo valor más alto y el primero que cumple la condición de ser positiva siempre, en el caso del ángulo múltiples distribuciones satisfacen pero como la distribución normal tenía un valor P del 93,79 % y es una distribución conocida, se seleccionó esta; finalmente, en el caso de la aceleración la distribución de Burr tipo III poseía un valor P de 99,28 % si bien había otras distribuciones con valores ligeramente mayor se optó por escoger esta porque ya la magnitud de la velocidad seguía esta distribución y simplifica la implementación.

En el caso de la vista exhaustiva se utilizó una mezcla entre un modelo de la señal no estadístico y el modelo estadístico previamente seleccionado de tal forma que se pueda obtener una gran cantidad de datos para poder realizar el análisis de frecuencia pero a la vez conservar la coherencia con el resto de mediciones; para esto se ajustó la amplitud de la señal de acuerdo al valor esperado según el modelo de la aceleración y para expresar el ruido de la medición a los parámetros del modelo se les suma una señal de ruido de tipo gaussiana.

Una vez obtenido el modelo, se implementó una API Web usando FastAPI con

dos endpoints, uno utilizado para la vista general y específica y otro para la vista exhaustiva. Ambos endpoints solicitan el nivel de daño el cual es un número entre 0 y 10 que indica el grado de daño del motor y solicitan el id del motor, el cual es un número entero que identifica el motor. Esta API se encuentra en un Host independiente en Digital Ocean y la accede el microservicio de Go para obtener los valores del modelo.

4.4. IMPLEMENTACIÓN DEL ANÁLISIS EN FRECUENCIA

Una de las herramientas mas importantes para tomar una decisión en cuanto al daño del motor es el análisis en frecuencia de la vibración, esto es porque permite obtener un mayor conocimiento del estado de los rodamientos y de cual puede ser la falla, en el caso de que exista alguna, que estos presentan. Para poder observar esta información, se procedió a realizar una transformación de dominio, mediante una FFT, para llevar las mediciones al dominio de la frecuencia y mediante una gráfica permitir el análisis.

Cabe destacar que este análisis y la generación de la gráfica son parte de la API del análisis exhaustiva, en el servidor Web y el resultado es visualizado mediante el cliente Web, en el endpoint correspondiente a la vista exhaustiva.

Para comenzar este proceso se debe de solicitar la información al servidor de sensorica, esto se hace mediante una **peticion Web GET** al endpoint correspondiente (DireccionIP/exhaustive?idMotor=identificador), posteriormente se examina el **header** de la respuesta para determinar si fue exitosa la solicitud, si esta no lo fue, no se puede generar este estudio por lo se envía una imagen de error asociada al cliente; en caso de que si se pueda proceder, se obtiene la información, proveniente en formato JSON, y mediante la librería numpy se obtiene un array con las mediciones, se procede a utilizar la transformada **FFT** de la misma librería, se convierte a una transformada unilateral, por estándar en la industria, y se gráfica mediante la librería **matplotlib** y se guarda en disco, para después ser enviada como un archivo estático por otro puerto del servidor Web.

4.5. IMPLEMENTACIÓN DE LA BASE DE DATOS

Como se ha mencionado anteriormente, se ha escogido una base de datos no relacional, específicamente MongoDB, dentro de ella se han desarrollado una base de datos llamada “tesis” y dos colecciones llamadas “MotorData” y “MotoresInDB” como se observa en 2; la primera colección se encarga de almacenar toda la información enviada por los sensores (en este caso el cliente simulado con los datos proporcionados por el modelo estadístico) y la segunda colección contiene una lista con los identificadores únicos de cada motor que tenga al menos un documento en la colección “MotorData”, es decir, hay información registrada de su actividad.

Cada colección tiene una estructura fija definida para facilitar el manejo y la consistencia de los documentos, esta es similar a un JSON y sus campos están explicados en las tablas 5 para “MotorData” y 6 para “MotorInDB”.

Tabla 5: Estructura de la colección MotorData

Elemento	tipo de dato	Descripción
_id	[]bytes	Elemento utilizado por MongoDB para identificar y facilitar la búsqueda de los documentos
IdMotor	string	Identificador único del motor.
Características	string	Descripción e información del motor.
IdSensor	[]uint64	lista de los identificadores-sensores que tiene conectado este motor.
Data	[]DataSensor	lista en forma de sub colección que contiene los resultados del sensor.
Time	int64	Estampa de tiempo, fecha y hora de la muestra en formato Unix.
Sub colección “DataSensor”		
IdSensorData	uint64	Identificador único del sensor que tomo la muestra.
Aceleración	float64	Muestra de aceleración medida en g .
VelocidadX	float64	Muestra de velocidad en el eje X.
VelocidadY	float64	Muestra de velocidad en el eje Y.
VelocidadZ	float64	Muestra de velocidad en el eje Z.

El símbolo “[]” indica que es un arreglo de ese tipo de datos.

Tabla 6: Estructura de la colección MotorInDB

Elemento	tipo	Descripción
_id	[]bytes	Elemento utilizado por MongoDB para identificar y facilitar la búsqueda de los documentos
IdMotor	[]string	Lista que contiene todos los identificadores únicos de los motores. Facilita búsqueda e implementación de los clientes Webs

El símbolo “[]” indica que es un arreglo de ese tipo de datos.

Cabe relatar que “uint64” hace referencia a un número natural de 64 bits y es usado en los identificadores de sensores ya que permite el uso de 64 bits (16 nibbles (grupos de cuatro)) los cuales son codificados de la forma expuesta en la tabla 7, para poder transmitir mas información y facilitar la escalabilidad del sistema en un futuro. Asimismo, se utiliza “float64” para representar a un número racional representado como punto flotante de 64 bits, esta es una unidad común porque permite maximizar la precisión en la medida.

Tabla 7: Estructura seguida en el identificador de los sensores para permitir y facilitar la escalabilidad del sistema

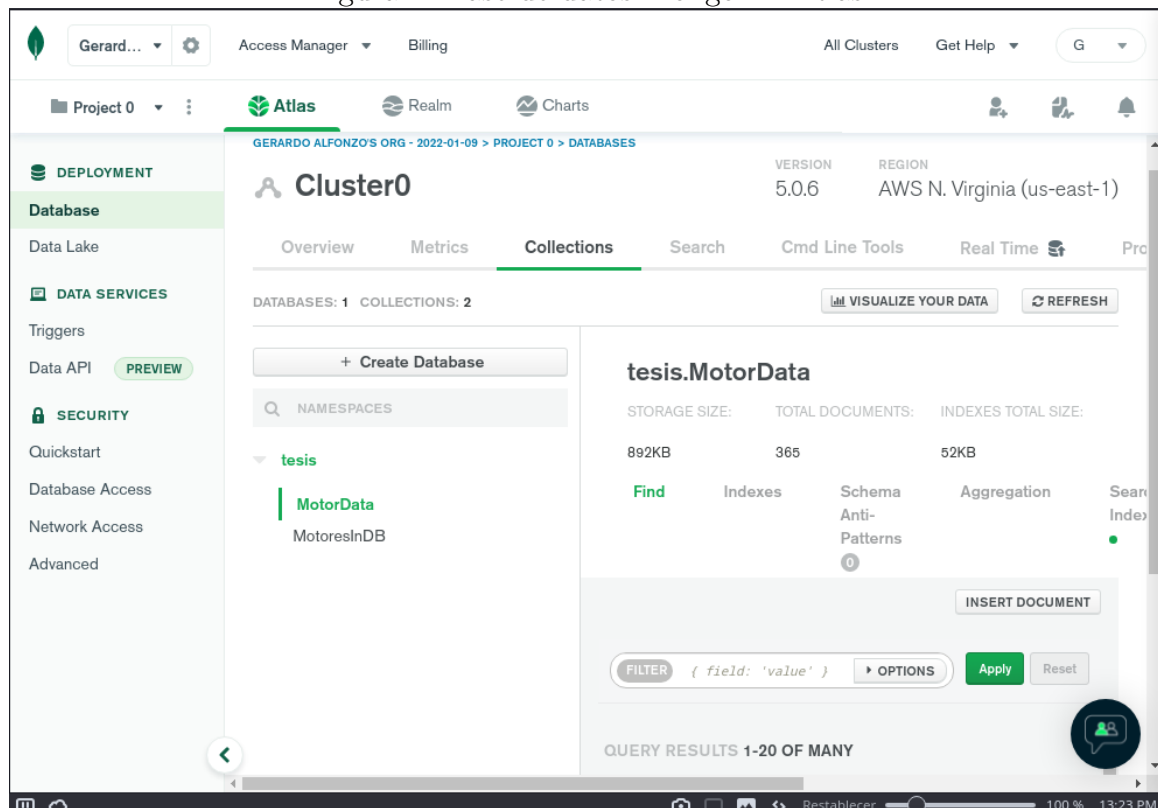
Tipo de sensor	Reservado	Ubicación	Serial
B_{15}	$B_{14}B_{13}$	B_{12}	$B_{11} \cdots B_0$

Campo	Descripción
tipo	tipo de sensor usado, Acelerómetro, Temperatura, etc. Con 0000b siendo Acelerómetro y 15 posibilidades adicionales.
Reservado	No se utilizan, son 0x00 siempre y se reservan para posibles expansiones y/o necesidades.
Ubicación	Posición con respecto al motor y acoples. Con: 0000b Lado con carga, 0001b Lado libre, 0010b \cdots 1111b disponibles para acoples y chumaceras.
Serial	número de fabricación del sensor, desde 0 hasta 2^{48} .

4.6. LLENADO DE LA BASE DE DATOS

Para el llenado con la información se desarrolló un Script en Golang, llamado “LlenarBBDD” , que se encarga de hacer las peticiones al modelo y a la base de

Figura 2: Base de datos MongoDB Atlas



Base de datos implementada en MongoDB Atlas.

datos de forma automática, realiza 120 peticiones, equivalentes a 120 días, con un aumento progresivo del nivel de daño cada 40 días-peticiones. Al ser un Script funciona mediante la terminal y utiliza las banderas expuestas en la tabla 8 para tomar la información. La utilización del mismo puede verse en 3

4.7. IMPLEMENTACIÓN DE LOS SERVIDORES

Los servidores son necesarios para recolectar la información de los sensores, establecer una comunicación full dúplex la cual permita obtener información a tiempo real del comportamiento del sensor, además de permitir toda la interacción Web que da la visibilidad. Dado que estas tareas pueden ser separadas y manejadas en forma de API se crearon 2 servidores, permitiendo de esta forma la división de la carga de trabajo y, por ende, disminuyendo los requerimientos mínimos del equipo en donde se monta cada servidor individual. Así mismo, esto permite una mayor escalabilidad y paralelismo, dado que en el caso de ser necesaria una ampliación en la capacidad de

Tabla 8: Banderas utilizables al momento de llamar el Script para llenar la información de la base de datos

Bandera	Descripción	Requerimiento
-i	Para especificar el Id del Motor.	Requerida.
-d	Para indicar el nivel de daño	Opcional, por defecto 1.
-c	Indicar las características e información del motor	Opcional, por defecto "Estado del motor".
-s1	Especificar el Id del sensor 1	Requerida.
-s2	Especificar el Id del sensor 2	Requerida.
-s3	Especificar el Id del sensor 3	Opcional, por defecto se omite.
-s4	Especificar el Id del sensor 4	Opcional, por defecto se omite.
-r	Reiniciar la base de datos si su valor es "true"	opcional, por defecto false.
-fi	Fecha de inicio, una cantidad de días contados hacia atrás desde la fecha actual	Opcional, por defecto 90
-ci	Cantidad de interacciones a ejecutarse	Opcional, por defecto, 90

cómputo se puede colocar otro equipo en vez de aumentar las capacidades del equipo ya existente. Este hecho permite disminuir los costos sustancialmente.

La implementación de estos 2 servidores da origen a un **servidor dedicado a sensorica**, desarrollado como un micro-servicio en el lenguaje de programación Go (también conocido como Golang) por las razones previamente expuestas, y a un **servidor dedicado al tratamiento Web** desarrollado con una combinación de Python y el Framework Django para el Backend y HTML-CSS-JavaScript con el Framework de React en un paradigma multipaginas de renderizado desde el servidor para el frontend-cliente Web.

4.7.1. Servidor dedicado a Sensorica

Este servidor fue realizado en Go por los motivos expuestos anteriormente y cumple la función de micro-servicio, se encarga de la recolección y comunicación con la red de sensorica, la cual es implementada por el cliente de la simulación y el modelo estadístico, así mismo envía la información relacionada con la vista exhaustiva (para

Figura 3: Llenado de la BBDD

```

[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$ ./LlenarBBDD
2022/03/11 13:24:33 Conexion exitosa con la BD
Faltan argumentos
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$ ./LlenarBBDD -i mal23 -d 7 -c "Motor de prueba
daño critico 1" -s1 45 -s2 23
2022/03/11 13:30:10 Conexion exitosa con la BD
2022/03/11 13:30:10 agregado a la BBDD
2022/03/11 13:30:11 El id de la impresion es
2022/03/11 13:30:11 ObjectID("622b8723bee96aea4aadd0e5")
2022/03/11 13:30:11 El id de la impresion es
2022/03/11 13:30:11 ObjectID("622b8723bee96aea4aadd0e6")
2022/03/11 13:30:12 El id de la impresion es
2022/03/11 13:30:12 ObjectID("622b8724bee96aea4aadd0e7")
2022/03/11 13:30:12 El id de la impresion es
2022/03/11 13:30:12 ObjectID("622b8724bee96aea4aadd0e8")
2022/03/11 13:30:13 El id de la impresion es
2022/03/11 13:30:13 ObjectID("622b8725bee96aea4aadd0e9")
2022/03/11 13:30:13 El id de la impresion es
2022/03/11 13:30:13 ObjectID("622b8725bee96aea4aadd0ea")
2022/03/11 13:30:14 El id de la impresion es
2022/03/11 13:30:14 ObjectID("622b8726bee96aea4aadd0eb")
2022/03/11 13:30:14 El id de la impresion es
2022/03/11 13:30:14 ObjectID("622b8726bee96aea4aadd0ec")
2022/03/11 13:30:15 El id de la impresion es
2022/03/11 13:30:15 ObjectID("622b8727bee96aea4aadd0ed")
2022/03/11 13:30:15 El id de la impresion es
2022/03/11 13:30:15 ObjectID("622b8727bee96aea4aadd0ee")
2022/03/11 13:30:16 El id de la impresion es

```

Llamados al Script para llenar la BBDD.

esta se requieren mediciones a tiempo real y por esto este servidor tiene una conexión full dúplex con los sensores); como se observa en la tabla 9

Cabe resaltar que "DireccionIP" hace referencia a la dirección en la que sera montado (Host) el sitio, en caso de un ambiente local, por ejemplo, es "localhost:8080"(este es el usado para las pruebas, cuando se cambia a producción se modifica por el del Host contratado)

El primer "Endpoint" (DireccionIP/sensormessage) realiza las siguientes tareas:

- Establecer una conexión Http2 con los solicitantes; para esto se intercambian, además de los paquetes e información necesarios para establecer el protocolo, unos mensajes que permiten identificar el motor y la cadena de sensores correspondientes a esta información.
- Posteriormente, se revisa si el motor tiene una conexión activa (no debe existir por unicidad de la información) y si ya se ha recibido información de este motor previamente; de no ser así, se agrega a la lista de motores de la que se posee

Tabla 9: Relación de punto de acceso y funcionalidad implementada en el servidor de sensorica

Dirección	Tarea realizada
DireccionIP/sensormessage	Se encarga del comportamiento, acceso e intercambio de información sensor-servidor. Es una comunicación bidireccional con Http2 (Https) y se intercambian por el canal establecido tanto la información de medición diaria, (después es subida a la base de datos) como la información de medición exhaustiva (es enviada al cliente que la solicitó).
DireccionIP/exhaustive ?idMotor=identificador	Se encarga de solicitar la información para la vista exhaustiva, el identificador único del motor que se solicita la data es enviado por el Url (?idMotor=identificador) .

información.

- En este punto el servidor se dedica a escuchar la llegada o solicitud de información. Esta puede ser del sensor con el que se estableció conexión (mediante un canal interno) o de la solicitud de información para una vista exhaustiva. Para cada caso se hace lo siguiente:
 - * Si es información, se verifica que venga en formato válido y se sube a la base de datos.
 - * Si es una solicitud de información, se verifica que la conexión con el motor sea la indicada y se solicita la información, se espera la respuesta y se envía por el mismo canal interno. Esta solicitud es hecha por el segundo Endpoint.
 - * No se dio ninguno de los casos, entonces el cliente se desconecta. Se procede a eliminar la conexión y la lista, mediante un procedimiento de cierre de conexión.
- Finalmente, existe un procedimiento de cierre de conexión ya sea por solicitud del cliente o por un error ocurrido.

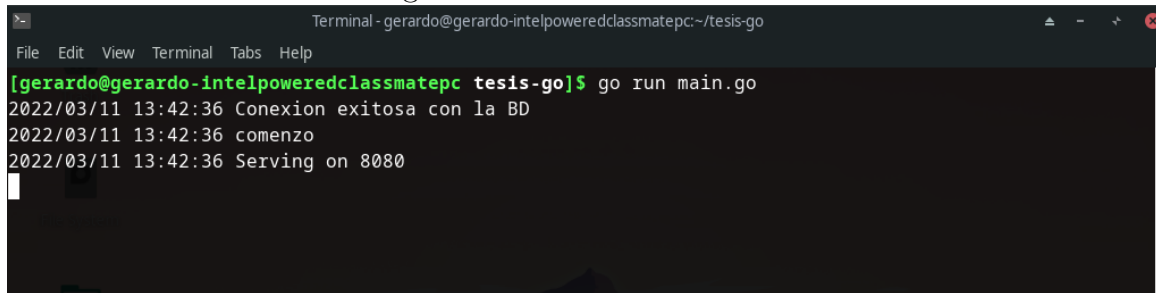
El segundo "Endpoint" (DireccionIP/exhaustive**?idMotor=identificador**) realiza

las siguientes tareas:

- Decodifica el URL enviado para obtener el parámetro (idMotor) y comprueba su existencia. En caso de error se envía un mensaje de petición incorrecta.
- Se comprueba que el motor solicitado exista en las conexiones actuales. Es importante resaltar que se refiere a la conexión bidireccional, ya que de caso contrario no se puede obtener información a tiempo real. Por esto, si la conexión es inexistente, se envía un mensaje de petición incorrecta, no se puede conectar al motor.
- Se solicita por un canal interno al otro Endpoint la información deseada y se espera su respuesta.
- Se envía la respuesta con estado de creado y la información.

El servidor corriendo es simplemente una terminal ejecutando un programa, puede ser visto en la imagen 4.

Figura 4: Servidor de sensorica



```
Terminal - gerardo@gerardo-intelpoweredclassmatepc:~/tesis-go
File Edit View Terminal Tabs Help
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$ go run main.go
2022/03/11 13:42:36 Conexion exitosa con la BD
2022/03/11 13:42:36 comenzo
2022/03/11 13:42:36 Serving on 8080
```

Inicio de ejecución del servidor de sensorica.

4.7.2. Servidor Web

Este servidor fue realizado en Python con el Framework Django por los motivos expuestos anteriormente y cumple la función de servidor Web, es el servidor principal ya que se encarga de enviar y solicitar información para crear la visualización del cliente, además de estudiar y crear las gráficas y enviar información al cliente Web para análisis. Sus conexiones son por el protocolo Http, y Https para la petición de

información exhaustiva con el microservicio de sensorica. y realiza las tareas expuestas en la tabla 10

Cabe resaltar que "DireccionIP" hace referencia a la dirección en la que será montado (Host) el sitio, en caso de un ambiente local, por ejemplo, es "localhost:8000" (este es el usado para las pruebas, cuando se cambia a producción se modifica por el del Host contratado)

Tabla 10: Relación de punto de acceso y funcionalidad implementada en el servidor Web

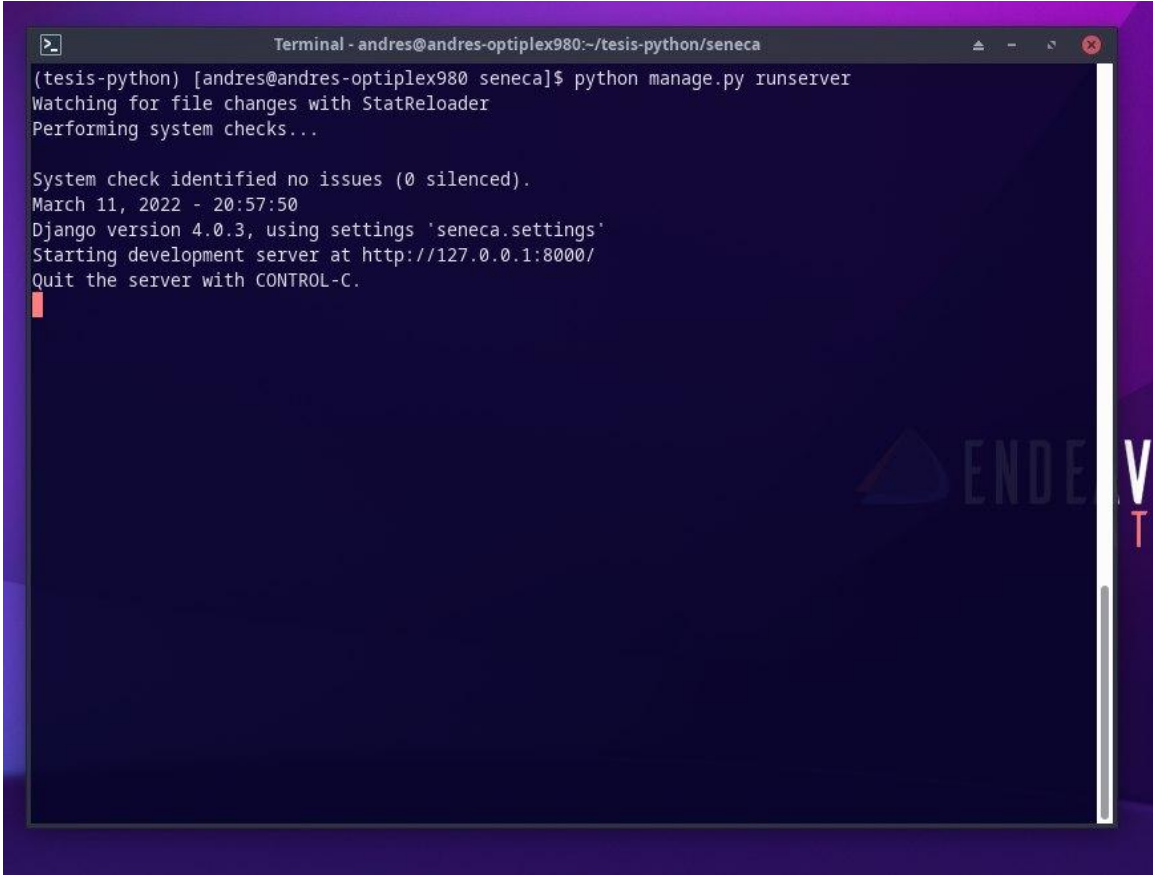
Dirección	Tarea realizada
DireccionIP/	Enviar el HTML necesario, mediante un Template de Django, para mostrar la vista general en el navegador.
DireccionIP/especifica ?IdMotor=identificador	Enviar el HTML necesario, mediante un Template de Django, para mostrar la vista especifica en el navegador.
DireccionIP/exhaustiva ?IdMotor=identificador	Enviar el HTML necesario, mediante un Template de Django, para mostrar la vista exhaustiva en el navegador.
DireccionIP/ static/...	Es una funcionalidad del servidor la cual permite el envío de archivos estáticos por solicitudes externas, ya sea por un requerimiento del HTML o del JavaScript.
DireccionIP/ get_data_motores	API que entrega una lista, formato JSON, con la ultima medición almacenada de cada sensor en la base de datos.
DireccionIP/ get_list_motores	API que entrega una lista, formato JSON, con todos los motores de los que se dispone información en la base de datos.
DireccionIP/get_especifica ?IdMotor=identificador	API que se encarga de devolver un JSON con toda la información en la base de datos referente al motor con el Id pasado en el Url, además de las direcciones, en el mismo servidor que serán entregadas mediante el Endpoint "DireccionIP/static/... ", que ocupan las gráficas históricas referente a ese motor.
DireccionIP/get_exhaustiva ?IdMotor=identificador	API que se encarga de devolver un JSON con toda la información en la base de datos referente al motor con el Id pasado en el Url, además de las direcciones, en el mismo servidor que serán entregadas mediante el Endpoint "DireccionIP/static/... ", que ocupan las gráficas históricas y del estudio de Fourier referente a ese motor.
DireccionIP/help	Endpoint que se encarga de enviar el HTML referente al manual de usuario.

Es decir, se realiza lo siguiente:

- Los Endpoint `DireccionIP/`, `DireccionIP/especifica` y `DireccionIP/exhaustiva` están encargados de retornar el código HTML correspondiente a cada una de las vistas para esto utilizan el sistema de Templates de **Django** el cual renderiza el código HTML desde el servidor. Los Endpoint para la vista específica y exhaustiva a su vez toman un parámetro GET llamado `IdMotor` el cual indica el motor que va a ser visualizado.
- `DireccionIP/get_data_motores`, es una **API** la cual retorna las últimas mediciones de todos los motores almacenados en la base de datos en formato JSON, para esto realiza una petición directamente a la base de datos, posteriormente coloca la respuesta en el formato correcto y la devuelve al cliente.
- `DireccionIP/get_list_motores`, es una **API** que retorna todos los identificadores de los motores guardados en la base de datos para esto solicita el campo en la base de datos que contiene la lista de todos los motores almacenados, la coloca en el formato correcto y la devuelve al cliente.
- `DireccionIP/get_especifica`, una **API** la cual requiere de un parámetro en la petición **GET** llamado `IdMotor`; este es utilizado para obtener todas las mediciones correspondientes a ese motor, después de obtenida estas, se realizan los histogramas se guardan como imágenes y se devuelve los Urls que direccionan las mismas junto a todas las mediciones.
- `DireccionIP/get_exhaustiva`, es una **API** cuyo funcionamiento es similar al `get_especifica` pero a su vez realiza una petición al microservicio de **sensorica** para obtener las mediciones de la vista exhaustiva, calcula la transformada rápida de Fourier y retorna un gráfico de la magnitud de esta en conjunto a los datos y los histogramas de la misma forma que la **API** anterior.
- `DireccionIP/help`, es un endpoint básico el cual envía, mediante templates de **Django** la información del manual de usuario.

El servidor corriendo es simplemente una terminal ejecutando un programa, puede ser visto en la imagen 5.

Figura 5: Servidor Web

A screenshot of a terminal window titled "Terminal - andres@andres-optiplex980:~/tesis-python/seneca". The terminal shows the command `python manage.py runserver` being executed. The output indicates that the system checks passed, the Django version is 4.0.3, and the development server is starting at `http://127.0.0.1:8000/`. The prompt `(tesis-python) [andres@andres-optiplex980 seneca]$` is visible at the top. The terminal background is dark blue with a faint "ENDE" logo and a red "V" on the right side.

```
(tesis-python) [andres@andres-optiplex980 seneca]$ python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
March 11, 2022 - 20:57:50
Django version 4.0.3, using settings 'seneca.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Inicio de ejecución del servidor Web.

4.8. IMPLEMENTACIÓN DE LOS CLIENTES

Un cliente puede ser definido como un equipo o software que se conecta a un servidor para obtener un beneficio, sea por poder de computo, para obtener una información dada o comunicarse con un programa que se ejecuta en el lado del servidor. Siguiendo esta definición, se crearon 2 clientes, uno para facilitar la inserción de datos en la simulación del sensor, y el cliente Web, el cual permite manejar el análisis y la monitorización de los sensores.

4.8.1. Cliente de Sensorica

Este fue elaborado en Go para facilitar la interconexión con el servidor y de igual forma aprovechar el paralelismo y la multiplataforma que el lenguaje ofrece. Se puede subdividir en 3 acciones fundamentales:

- GUI: Es la interfaz gráfica de usuario, utiliza el Framework Fyne por facilidades de diseño y es una ventana que permite ingresar datos equivalentes a las características del motor y conectar con el servidor (además de especificar en qué dirección está el motor), y posee una ventana de **log** en la cual se comunica al usuario acciones como la conexión exitosa y el envío de información al servidor. esta puede ser vista en la figura 6 y como se observa, permite especificar:

1. Dirección Ip: Lugar a conectar.
2. Id Motor: Identificador único del motor.
3. Potencia: Información adicional, opcional.
4. Nivel de daño: numero entre 1 y 10 que determina los posibles valores del modelo.
5. Información: Información adicional, pensado para comentarios, opcional.
6. Sensores: lista de 1 a 5 posibles sensores, que representan los identificadores únicos que tienen los sensores asociados al motor.

Cabe resaltar que a ser una GUI es el proceso principal, las demás tareas son realizadas de forma paralela.

- Conexión al servidor: Es un protocolo que ocurre cada vez que se presiona el botón de **conectar**, se encarga de intercambiar información con el servidor para poder conectarse (bajo el protocolo Https) y envía los datos de qué motor se va a conectar (simular) y qué sensores tiene asociados, espera una aprobación de conexión (para que no exista un motor con el mismo id enviando información), obtiene las características especificadas en la GUI y comienza un proceso de envío-recibo de información. Este es un proceso que se ejecuta paralelamente a la GUI y se inicia cada vez que se presiona el botón de **conectar**, Si había un proceso previo y se vuelve a presionar **conectar** finaliza el anterior y comienza uno nuevo.
- Comunicación con el servidor: Es un proceso de envío bidireccional de información, esta rutina se inicia cuando la conexión al servidor es completada exitosamente y se subdivide en 3 procesos que, a su vez, corren paralelamente

(se toma el proceso de conexión al servidor y se crean 2 hijos para un total de 4 procesos paralelizados si se incluye la GUI). Estos procesos son:

1. **timer**, Es un proceso que se encarga de cronometrar cada cuánto se va a mandar un mensaje de información con los datos correspondientes a una medición normal al servidor.
2. **listen**, Es un proceso que se encarga de verificar si hay una solicitud de información, ya sea del subprocesso **timer** (como un mensaje normal) o del servidor para solicitar información de la vista exhaustiva (la cantidad de información enviada es sustancialmente diferente) o para la terminación del proceso e informa a **handler** qué se debe hacer.
3. **handler**, Se encarga de realizar la tarea pedida por **listen**, al enviar la información solicitada al servidor, enviando 2 mensajes; uno con el tipo de información que se envía y otro con la información. Esto fue establecido como una especie de protocolo para dar mayor seguridad y a la vez facilitar el intercambio de información con el servidor.

Cabe resaltar que estos procesos son asíncronos y no sufren prelación entre ellos.

4.8.2. Cliente Web

Este cliente es el mas conocido, ya que es el que permite que se muestre la información en el navegador Web. Está constituido por las vistas general, específica y exhaustiva, cada una de ellas representa una página Web separada y todas fueron construidas utilizando el Framework de JavaScript **React** y con HTML específicamente un Template de Django y CSS para dar estructura básica y estilo respectivamente.

Se optó por utilizar un estilo multi páginas con renderizado de lado servidor (específicamente del servidor Web) por la necesidad de los cálculos avanzados y gráficas que se deben realizar, además de los llamados importantes e interconexiones

Figura 6: GUI del cliente de sensorica

The screenshot shows a window titled 'Motor' with the following sections:

- Conección:** IP: localhost:8080 (with a checkmark), a 'conectar' button.
- Motor:** ID: 2451 (with a checkmark), Potencia(HP): 1.
- Damage level:** A slider set at 0%.
- Información:** A dropdown menu showing 'Datos del motor' (with a checkmark).
- Sensores:** A table with 5 rows:

Sensor	Value	Status	Description
s1:	1	✓	Lado Libre
s2:	2	✓	Lado Carga
s3:	0	✓	chumacera-accesorio
s4:	0	✓	chumacera-accesorio
s5:	0	✓	chumacera-accesorio
- Log:** Displays 'Hi!'.

Interfaz de usuario para la conexión del cliente de sensorica.

con la API del servidor de Go y de las BBDD. Esto difiere con el paradigma tradicional de React (monopágina de renderizado en servidor) pero permite optimizar recursos y facilita expansiones a futuro.

Su estructura viene dada por las 3 páginas o sub aplicaciones que permiten:

- General: conocer el estado general de un grupo de motores, indicando en código de colores (verde, amarillo, rojo) el nivel de daño que posee un motor. Este nivel es determinado por las muestras mas actuales de la información del motor y unos valores parámetros proporcionados en conjunto con los datos de los cuales se elaboró el modelo estadístico.

Cabe resaltar que estos valores son una extrapolación empírica de la vibración en una planta específica, es configurable y puede variar dadas las características

propias de cada instalación. Esto es debido a que las bases utilizadas en la instalación, los soportes, entre otros factores **causados por ignorar las normas de instalación** afectan las medidas.

- Específica: permite el estudio del estado de un motor específico, es enviado el parámetro que lo identifica en el Url, al servidor. Permite observar una gráfica histórica y una tabla exportable a Excel de sus características y evolución en el tiempo, siempre y cuando se tenga medición de ese periodo en la base de datos, asimismo permite solicitar la vista exhaustiva del mismo motor si es requerido un nivel mayor de análisis.
- Exhaustiva: esta vista incluye todo lo anterior de la vista específica, con la diferencia de que permite regresar al menú general en vez de hacer una vista exhaustiva; además, realiza un análisis en frecuencia del estado a tiempo real del motor. Este análisis se puede realizar si se tiene acceso en tiempo real con el motor, es decir, el servidor de **sensorica** debe tener una conexión con un sensor que se encargue de monitorear el respectivo motor permitiéndole así solicitar la información necesaria para hacer el análisis.

Cabe resaltar que esta acción es tratada como una vista aparte ya que tiene un peso computacional relativamente alto asociado y, por ende, además de consumir recursos tiene una duración de carga de algunos segundos que deteriora la experiencia de usuario, por esto obtiene solamente por una solicitud explícita.

4.9. COMPROBACIÓN DE LOS RESULTADOS

Como se ha mencionado anteriormente, el trabajo consta de una gran cantidad de puntos que han sido trabajados de forma modular con la finalidad de maximizar la escalabilidad y de facilitar al máximo posible el realizar pruebas a los módulos y a los acoples realizados. La comprobación debe de tomar en consideración 2 factores, los resultados técnicos, es decir funcionalidades de la aplicación las cuales, como se mencionó anteriormente, no se automatizaron por factores tiempo y cantidad de pruebas y, los resultados finales, es decir la capacidad de la herramienta de facilitar el trabajo, mediante las páginas generales, específicas y exhaustivas.

4.9.1. Comprobación de resultados a nivel técnico

Las fallas internas de los servidores, o de procesos de comunicación con la base de datos, conllevan a la finalización de la ejecución del proceso (aplicación) correspondiente; esto se hace para poder reiniciar el sistema y agregar el error o falla catastrófica a un archivo (log) que sirve para seguir el comportamiento del servidor.

En el caso del servidor de sensorica, estos errores son manejados de forma manual y se utiliza la característica de Go **"panic"** para detener la ejecución, mas específicamente un paquete de la librería estándar **"log"** con la función **"log.Fatalf"**. Por otro lado, el Framework de Django se encarga de esto, si el error es manejable; algún problema en una petición, o algo no catastrófico, arroja una excepción y un error 500, fallo interno en el servidor, y finaliza la petición; en el caso de no poder ser recuperado finaliza el proceso.

La finalización del servidor es una práctica relativamente común ya que permite el reinicio del proceso, esto se hace mediante una configuración interna al momento de hacer **deploy** en el servidor, como se observa en la figura 7 es la configuración recomendada por DigitalOcean para un reinicio continuo, en caso de falla, a intervalos de 5 segundos.

Figura 7: Configuración de proceso en Linux para reinicio automático de procesos

```
/lib/systemd/system/goweb.service

[Unit]
Description=goweb

[Service]
Type=simple
Restart=always
RestartSec=5s
ExecStart=/home/user/go/go-web/main

[Install]
WantedBy=multi-user.target
```

digitalocean (s.f.).

Por esta razón, las pruebas mas considerables son interconexiones y configuraciones de seguridad y problemas de datos. es decir:

- Prueba de no unicidad de la información, al haber múltiples clientes de sensorica enviando información sobre el mismo motor.

Para esta posibilidad, se optó por una configuración que rechaza el segundo intento de conexión, es decir, se rechaza cualquier intento de conexión de motor con una conexión establecida previamente (que todavía exista), y su implementación se observa en la figura 8.

Figura 8: Prueba de error, no unicidad de información

```

[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$ go run cliente.go
2022/03/06 16:10:05 localhost:8080 Datos del moto
2022/03/06 16:10:05 2451
2022/03/06 16:10:05 1
2022/03/06 16:10:05 0
2022/03/06 16:10:05 0
valor: 0 1
2022/03/06 16:10:05
valor: 1 1000000000000000
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$

[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$ go run cliente.go
2022/03/06 16:10:09 localhost:8080 Datos del moto
2022/03/06 16:10:09 2451
2022/03/06 16:10:09 1
2022/03/06 16:10:09 0
2022/03/06 16:10:09 0
valor: 0 1
2022/03/06 16:10:09
valor: 1 1000000000000000
2022/03/06 16:10:09 Failed login 2: <nil>
exit status 1
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$

[gerardo@gerardo-intelpoweredclassmatepc ~]$ cd tesis-go
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$ go run main.go
2022/03/06 16:08:53 Conexion exitosa con la BD
2022/03/06 16:08:53 comenzo
2022/03/06 16:10:06 Serving on 8080
2022/03/06 16:10:06 [::1]:53620] Trata de conectarse: 2451
2022/03/06 16:10:06 login sucesed
2022/03/06 16:10:06 agregado a la BD00
2022/03/06 16:10:06 El id de la impresion es
2022/03/06 16:10:06 ObjectID("6225151e8aee853f87604a42")
2022/03/06 16:10:09 [::1]:53622] Trata de conectarse: 2451
2022/03/06 16:10:09 http2: panic serving [::1]:53622: runtime error: invalid memory address or
nil pointer dereference
goroutine 209 [running]:
net/http.(*http2serverConn).runHandler.func1()
/usr/lib/go/src/net/http/h2_bundle.go:5842 +0x125

```

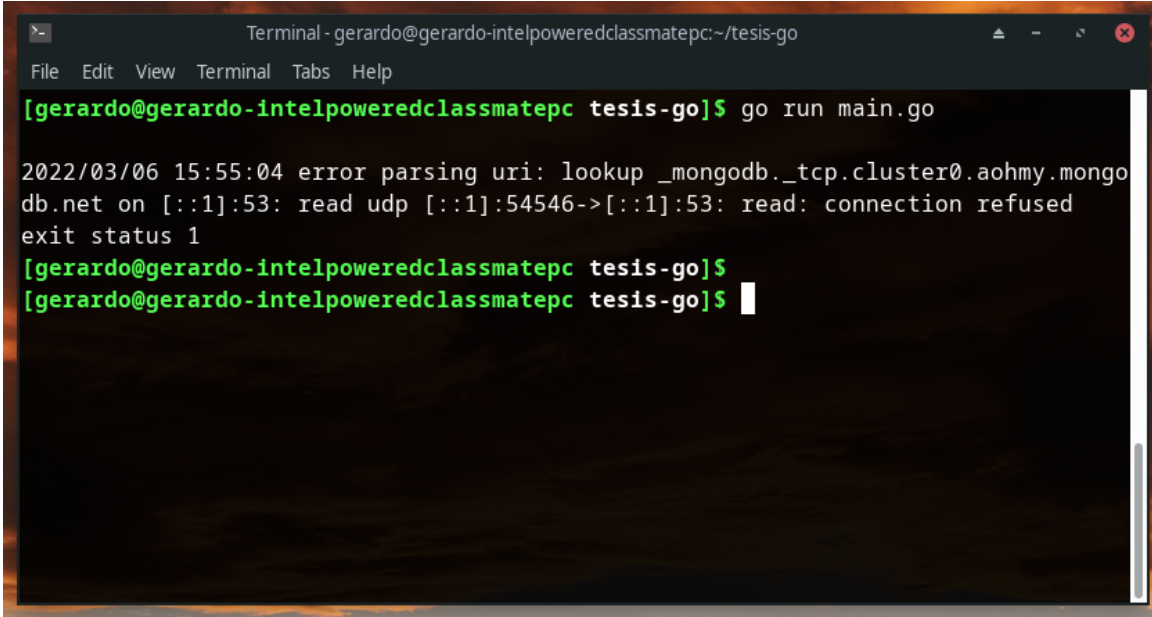
logs: Izquierda cliente conectado, Centro cliente Rechazado, Derecha. Servidor.

- Prueba de falla en la conexión de la base de datos.

Cada servidor toma esta falla de una forma específica, en el caso del servidor de sensorica, esta es considerada una falla catastrófica, por lo que se finaliza la ejecución, como se observa en la figura 9.

Por otro lado, el servidor Web arroja una excepción, la cual es manejada internamente por Django y resuelta de forma que la petición es respondida con un error 500, falla interna del servidor, este error en la respuesta dada es manejado por el cliente y se observa en la figura 10 .

Figura 9: Prueba de error, no BBDD Sensorica



```

Terminal - gerardo@gerardo-intelpoweredclassmatepc:~/tesis-go
File Edit View Terminal Tabs Help

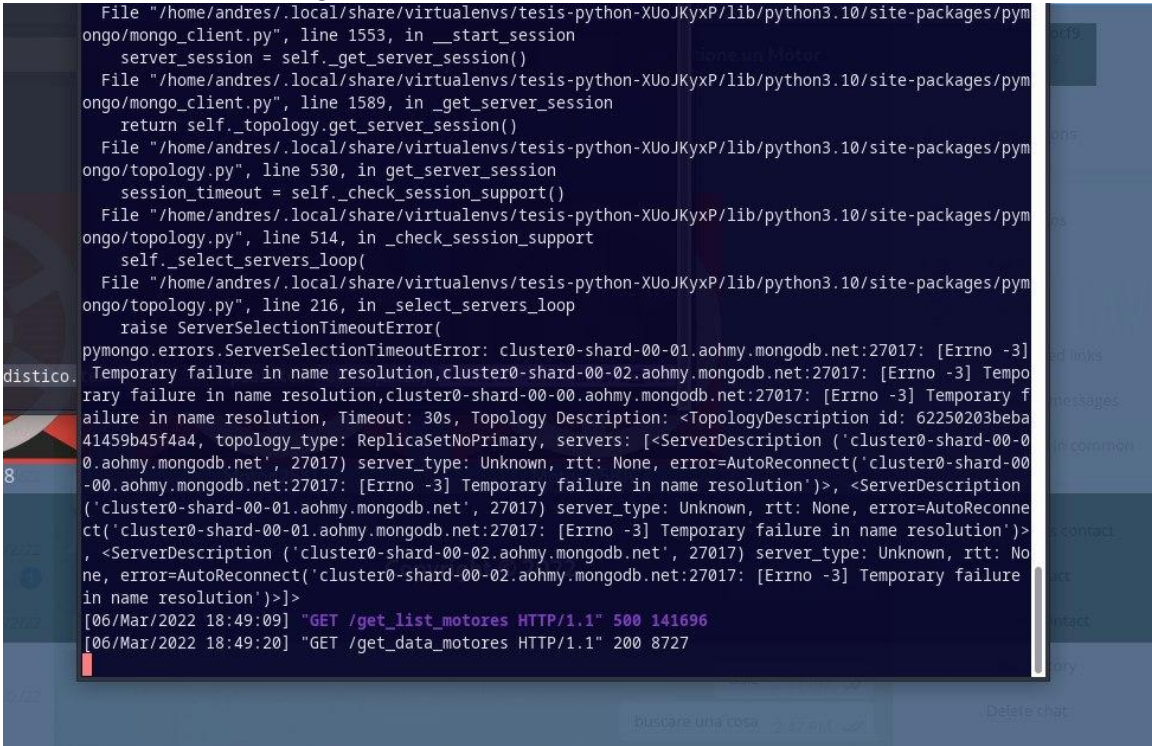
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$ go run main.go

2022/03/06 15:55:04 error parsing uri: lookup _mongodb._tcp.cluster0.aohmy.mongod
b.net on [::1]:53: read udp [::1]:54546->[::1]:53: read: connection refused
exit status 1
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$
[gerardo@gerardo-intelpoweredclassmatepc tesis-go]$

```

El sistema no inicia por no poder conectarse a la BBDD.

Figura 10: Prueba de error, no BBDD Web



```

File "/home/andres/.local/share/virtualenvs/tesis-python-XUoJKyxP/lib/python3.10/site-packages/pym
ongo/mongo_client.py", line 1553, in __start_session
    server_session = self._get_server_session()
File "/home/andres/.local/share/virtualenvs/tesis-python-XUoJKyxP/lib/python3.10/site-packages/pym
ongo/mongo_client.py", line 1589, in _get_server_session
    return self._topology.get_server_session()
File "/home/andres/.local/share/virtualenvs/tesis-python-XUoJKyxP/lib/python3.10/site-packages/pym
ongo/topology.py", line 530, in get_server_session
    session_timeout = self._check_session_support()
File "/home/andres/.local/share/virtualenvs/tesis-python-XUoJKyxP/lib/python3.10/site-packages/pym
ongo/topology.py", line 514, in _check_session_support
    self._select_servers_loop()
File "/home/andres/.local/share/virtualenvs/tesis-python-XUoJKyxP/lib/python3.10/site-packages/pym
ongo/topology.py", line 216, in _select_servers_loop
    raise ServerSelectionTimeoutError(
pymongo.errors.ServerSelectionTimeoutError: cluster0-shard-00-01.aohmy.mongodb.net:27017: [Errno -3]
Temporary failure in name resolution, cluster0-shard-00-02.aohmy.mongodb.net:27017: [Errno -3] Tempo
rary failure in name resolution, cluster0-shard-00-00.aohmy.mongodb.net:27017: [Errno -3] Temporary f
ailure in name resolution, Timeout: 30s, Topology Description: <TopologyDescription id: 62250203beba
41459b45f4a4, topology_type: ReplicaSetNoPrimary, servers: [<ServerDescription ('cluster0-shard-00-0
0.aohmy.mongodb.net', 27017) server_type: Unknown, rtt: None, error=AutoReconnect('cluster0-shard-00
-00.aohmy.mongodb.net:27017: [Errno -3] Temporary failure in name resolution')>, <ServerDescription
('cluster0-shard-00-01.aohmy.mongodb.net', 27017) server_type: Unknown, rtt: None, error=AutoReconne
ct('cluster0-shard-00-01.aohmy.mongodb.net:27017: [Errno -3] Temporary failure in name resolution')>
, <ServerDescription ('cluster0-shard-00-02.aohmy.mongodb.net', 27017) server_type: Unknown, rtt: No
ne, error=AutoReconnect('cluster0-shard-00-02.aohmy.mongodb.net:27017: [Errno -3] Temporary failure
in name resolution')>]>
[06/Mar/2022 18:49:09] "GET /get_list_motores HTTP/1.1" 500 141696
[06/Mar/2022 18:49:20] "GET /get_data_motores HTTP/1.1" 200 8727

```

El sistema continua ejecución y resuelve el error con un error 500.

- Prueba de solicitud de vista exhaustiva a un motor sin conexión establecida.

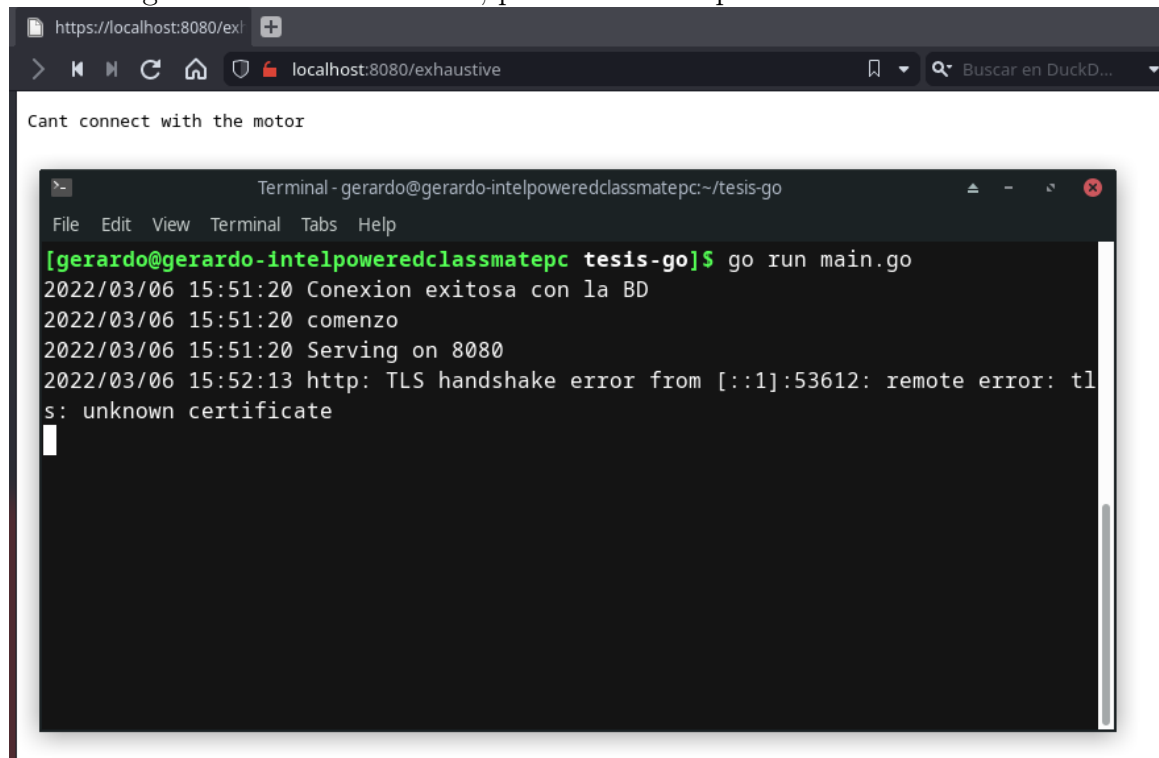
Este caso corresponde al servidor de sensorica; al no haber una conexión con el sensor establecida no se pueden tomar las mediciones para una vista exhaustiva,

estas deben ser en tiempo real, por lo tanto se responde con una error en la respuesta como se observa en la imagen 11.

- Prueba de petición no completada en el cliente Web, información invalida (motor no existente) o error del servidor Web.

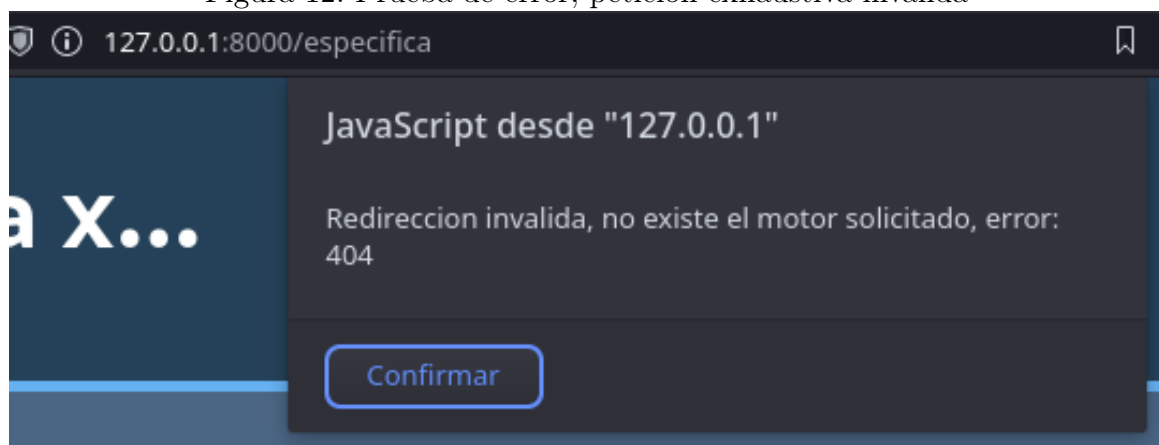
En este caso se muestra al cliente un error por pantalla, como ventana emergente, y se redirige a la vista general como se observa en la figura 12.

Figura 11: Prueba de error, petición no completada en el cliente Web



Solicitud desde el navegador, para facilitar vista, al servidor sin motores conectados.

Figura 12: Prueba de error, petición exhaustiva inválida



Solicitud no completada en el cliente Web.

4.9.2. Comprobación de resultados finales

Una vez terminado todo el proyecto se debieron rectificar que los requerimientos solicitados, en términos de parámetros, sean cumplidos al igual que las gráficas resultantes, para las históricas como para el análisis en frecuencia, tengan coherencia, dado esto se procedió de la siguiente forma:

- Corroboración del análisis en nivel de daño básico (vista general) que se ajuste a los parámetros requeridos por el cliente.

Para esta prueba se ingresaron 5 motores a la BBDD cuya información fue estrictamente configurada para cumplir los 3 estados de daño, y los 2 valores frontera. Consiguiendo de esta forma cubrir todas las posibilidades, este resultado puede verse en la figura .

- Corroboración de las gráficas históricas.

Esta verificación es conseguida mediante la comparación con los valores históricos, igual que para la prueba anterior, se ingreso un motor con valores de medición fáciles de seguir para corroborar la legibilidad y precisión de la gráfica, como se puede observar en la figura .

- Corroboración de la gráfica de la transformada de Fourier.

En este caso se procedió a comparar la gráfica obtenida mediante el algoritmo de Python con una obtenida mediante el software Octave (versión código abierta de Matlab), los resultados pueden verse en la figura .

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Tras la exposición de los resultados plasmados en el Capítulo IV, se pueden plantear las siguientes conclusiones:

- La investigación realizada ha sido de tipo aplicada y su naturaleza la clasifica como proyecto especial. Ésta ha consistido en la creación de una herramienta computacional para el análisis de la vibración en motores eléctricos mediante la implementación de dos servidores, uno de estilo microservicio, y un cliente Web, la cual puede ser alimentada mediante simulaciones, como es el caso del modelo estadísticos y el sistema de sensorica, o mediante mediciones a tiempo real de cualquier sistema que respete las estructuras de datos empleadas.
- Se deciden implementar múltiples servicios, cada uno escrito con un lenguaje de programación que satisface los requerimientos de velocidad, paralelismo y robustez necesarios para satisfacer los requerimientos de uso, además de una fácil adecuación y modificación a las necesidades, de múltiples clientes.
- Existe una gran cantidad de lenguajes de programación en la actualidad diseñados para satisfacer una gran cantidad de tareas en múltiples campos, siendo Go y Python herramientas de gran poder y fácil implementación al momento de realizar servidores y API, además de la creación de Scripts para la automatización de procesos; de igual forma Python es increíblemente polifacético y cuenta con una gran cantidad de librerías y Frameworks especialmente en el área de la ciencia de datos y estadística; sin embargo en términos de programación Web a nivel cliente, es casi la única posibilidad el

utilizar HTML, CSS, JavaScript, presentando el ultimo la mayor variedad de Frameworks para agilizar el trabajo, siendo React uno de los mas grandes en la industria.

- Se escoge implementar dos modelos estadísticos, uno para entregar el equivalente a mediciones diarias y otro para mediciones continuas a tiempo real necesarias para hacer un análisis en frecuencia; de esta forma se utilizaron distribuciones de probabilidad para representar 3 variables de interés, velocidad vertical, horizontal y aceleración, expresadas de forma independiente como magnitud y ángulo de la velocidad y aceleración con las dos primeras siendo la representación en coordenadas polares; las distribuciones que satisfacen de mejor manera estas variables fueron la distribución Burr de tipo III para la magnitud de velocidad y aceleración, y la distribución normal para el ángulo. De esta forma, se obtienen las mediciones diarias y con un modelo de señal no estadísticos ajustado a la amplitud dada por el modelo de la aceleración y una gaussiana para simular ruido, se obtienen las mediciones continuas.
- Para facilitar la accesibilidad al modelo se implementó una API Web con el Frameworks FastAPI y dos endpoints que entregan la información especificada en el Url vía parámetros de configuración, nivel de daño (un numero del 0 al 10) y el identificador único del motor que tiene asociado.
- Una vez obtenidas las salidas del modelo estadísticos, se procede con la implementación del análisis en frecuencia, esto se logra mediante una transformada rápida de Fourier (FFT) lo que permite llevar la señal discretizada (mediciones a tiempo real) al dominio de la frecuencia y posteriormente graficarla; todo esto se logra mediante las librerías **numpy** y **matplotlib** de Python.
- Se utilizo una base de datos no relacional, **MongoDB**, en forma de microservicio con la empresa **MongoDB Atlas** la cual permite la creación de un **cluster** gratuito para finalidades de practicas y estudios, capaz de almacenar una cierta cantidad de información e implementar diversas bases de datos y colecciones para las mismas, siempre que no se exceda este limite; esta base de datos llamada **tesis** contiene 2 colecciones, una se encarga de almacenar la información de

las mediciones diarias y la otra de almacenar los identificadores únicos que representan a los motor de los que se posee información.

- La automatización de procesos es fundamental para agilizar las tareas y la forma mas sencilla y óptima de hacerlo es mediante un Scripts; para llenar la base de datos se utilizo uno escrito en Go el cual acepta configuración al momento de la ejecución vía parámetros, siendo obligatorios la especificación de los identificadores únicos correspondientes al motor y los primeros 2 sensores.
- Se crearon 2 microservicios como servidores, el primero llamado **servidor de sensorica** se encarga de la comunicación con los sensores, simulados mediante el **cliente de sensorica** y el **modelo estadísticos**, y la entrega de la información tomada a tiempo real necesaria para el análisis en frecuencia. Por otro lado, el otro microservicio, **servidor Web**, se encarga de satisfacer las peticiones Web al enviar el HTML-CSS-JavaScript necesario para mostrar la pagina, además de los archivos estáticos e información requeridos por los mismos para mostrar el estado de los motores dependiendo del nivel de análisis solicitado.
- Para facilitar la utilización del modelo estadísticos y la conexión con el microservicio de **sensorica** se implemento un **cliente de sensorica** escrito en Go el cual permite especificar las características inherentes al motor (Identificador del motor y de los sensores, características, etc) y al daño existente en el mismo (en una escala del 1 al 10), además permite especificar la dirección IP, o puerto local, al cual se conectara.
- Se crearon 3 vistas para el cliente Web, estas son:
 1. **General**, tiene el menor grado de estudio ya que permite conocer el estado de todos los motores registrados en la BBDD, esto se consigue mediante paginación en una ventana, esta muestra 12 motores por vez y permite mover el índice para mostrar los siguientes 12, regresar e ir automáticamente al inicio, además existe una ventana de búsqueda que permite especificar el identificador único del motor del cual se quiere obtener la información.

2. **específica**, tiene un grado de estudio intermedio dado que permite conocer la evolución histórica del motor mediante gráficas y una tabla exportable a Excel; adicionalmente permite solicitar la vista exhaustiva.
 3. **exhaustiva**, tiene el mayor grado de estudio ya que solicita mediciones a tiempo real de la aceleración del motor y las descompone en frecuencia, permitiendo observar una gráfica de las frecuencias y magnitudes de las mismas en las que vibra el motor; adicionalmente muestra toda la información de la vista específica.
- Dado el tiempo de desarrollo y la complejidad del sistema no se pueden escribir **test** automatizados, sin embargo es absolutamente necesario el realizar pruebas a un software para garantizar su funcionamiento, por esta razón se probó manualmente el sistema en los puntos claves e interconexiones, es decir, se comprobó:
 1. La unicidad de la información.
 2. Fallas de conexión con el microservicio de base de datos.
 3. Solicitud de información a tiempo real a un motor sin conexión establecida.
 4. Peticiones invalidas en el servidor Web
 - Una vez establecido el sistema, se procedió a comprobar que cumpliera los requerimientos mínimos solicitados por el cliente, es decir, se verificaron la clasificación en los distintos niveles de daño de acuerdo a las especificaciones del cliente, se corroboró la coherencia y legibilidad de las gráficas históricas y de la gráfica del análisis en frecuencia.

5.2. Recomendaciones

- Debido al costo que implica no se realizaron placas de adquisición de datos y tampoco se obtuvo data en tiempo real de motores eléctricos para la elaboración de este trabajo, pero el mismo puede ser complementado implementando dichos dispositivos.

- Para su uso empresarial la aplicación requiere de un sistema de autenticación y de seguridad para que solo los usuarios con los privilegios correctos puedan acceder a la información, esto probablemente requiera el uso de otra base de datos preferiblemente una base de datos relacional.
- Un sistema de notificaciones configurable por el usuario, para observar si algún motor recibe una medida fuera de los valores normales puede ser una gran ayuda al operador de la aplicación en el diagnóstico de fallas de forma temprana.
- Debido a la complejidad de las conexiones HTTP2 una alternativa es el uso de Websockets los cuales permiten una conexión bidireccional sin la complejidad del nuevo protocolo pero posee su propia serie de desventajas las cuales tienen que ser evaluadas con los requerimientos de la aplicación. Otra alternativa aún no disponible pero en un futuro cercano posiblemente lo este, es HTTP3 la cual está siendo discutida por el comité del estándar HTTP para corregir las fallas de la última versión.
- Las pruebas fueron realizadas de forma manual pero una decisión que puede minimizar el tiempo de depuración y evitar las fallas en producción es el uso de pruebas automatizadas pero eso implica un coste en el tiempo de desarrollo dedicado al diseño de las pruebas por lo tanto la elección de usarlas o no dependerá de la experiencia previa del programador con este tipo de metodología de desarrollo.
- El ecosistema de librerías y Frameworks frontend es muy cambiante por lo tanto la elección de las herramientas debe ser realizada de acuerdo a las tecnologías más populares al momento del desarrollo.
- Si bien no se llegó a utilizar Julia sigue siendo un lenguaje con mucho potencial tratando de incorporar ventajas de lenguajes interpretados como Python y lenguajes compilados como Fortran. Su principal desventaja es lo joven que es pero a futuro puede llegar ser uno de los lenguajes más dominantes en el campo científico.

REFERENCIAS BIBLIOGRÁFICAS

- AddAppTo. (2015). *¿Qué es un sistema web?* [Online; fecha de consulta: 31/7/2021]. <http://www.addappto.com/que-es-un-sistema-web/>
- Aliat Universidades, U. E. 2. (s.f.). *LENGUAJES DE PROGRAMACIÓN: ¿QUÉ SON Y PARA QUÉ SIRVEN?* [Online; fecha de consulta 24/2/2022]. <https://etac.edu.mx/blog-etac/index.php/lenguajes-de-programacion/>
- Arias, F. G. (2006). *EL PROYECTO DE INVESTIGACIÓN, Introducción a la metodología científica* [Online; fecha de consulta 4/3/2022]. EDITORIAL EPISTEME, C.A. https://www.researchgate.net/publication/273441897_El_Proyecto_de_Investigacion_Introduccion_a_la_metodologia_cientifica_5ta_Edicion_Premio_Nacional_2006
- Çağlar Ramazan, İ. S., & Serhat, Ş. (2014). Statistical Wiener process model for vibration signals in accelerated aging processes of electric motors. [fecha de consulta: 1/8/2021]. *Journal of Vibroengineering*, 16(12).
- community, G. (s.f.). *git -fast-version-control* [Online; fecha de consulta 2/3/2022]. <https://git-scm.com>
- Corder, G. W., & Foreman, D. I. (2009). *Nonparametric Statistics For Non-Statisticians* [online; fecha de consulta 12/3/2022]. Wiley.
- DigitalOcean. (s.f.). *DigitalOcean - The developer cloud* [Online; fecha de consulta 6/3/2022]. <https://www.digitalocean.com>
- digitalocean. (s.f.). *how to deploy a go web application using nginx on ubuntu 18.04* [Online; fecha de consulta 6/3/2022]. <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-go-web-application-using-nginx-on-ubuntu-18-04>
- Eaton., J. W. (s.f.). *About* [Online; fecha de consulta 27/2/2022]. <https://www.gnu.org/software/octave/about>
- Education, I. C. (s.f.). *Application Programming Interface (API)* [Online; fecha de consulta 2/3/2022]. <https://www.ibm.com/cloud/learn/api>

- Escobar, G. (s.f.). *Lenguajes compilados e interpretados* [Online, blog; fecha de consulta 24/2/2022]. <https://blog.makeitreal.camp/lenguajes-compilados-e-interpretados/>
- Facebook Open Source, M. P. (s.f.). *React A JavaScript library for building user interfaces* [Online; fecha de consulta 24/2/2022]. <https://reactjs.org>
- Foundation, D. S., & individual contributors. (s.f.). *Django* [Online; fecha de consulta 27/2/2022]. <https://www.djangoproject.com/start/overview/>
- Foundation, P. S. (s.f.). *General Python FAQ* [Online; fecha de consulta 27/2/2022]. <https://docs.python.org/3/faq/general.html>
- Fraden, J. (2003). *Handbook of Modern Sensors: Physics, Designs, and Applications* [Online; fecha de consulta: 18/6/2021]. Springer.
- fyne.io. (s.f.). <https://fyne.io>
- Girdhar, P., & Scheffer, C. (2004). 5 - Machinery fault diagnosis using vibration analysis [Online; fecha de consulta: 18/6/2021]. En P. Girdhar & C. Scheffer (Eds.), *Practical Machinery Vibration Analysis and Predictive Maintenance* (pp. 89-133). Newnes. <https://doi.org/https://doi.org/10.1016/B978-075066275-8/50005-9>
- Golang.org. (s.f.). *Documentation* [Online; fecha de consulta 24/2/2022]. <https://go.dev/doc/>
- Hernández, M. T. (1998). Manual de Trabajos de Grado de Especialización y Maestría y Tesis Doctorales [fecha de consulta: 9/6/2021]. *Universidad Pedagógica Experimental Libertador*, 14-15. <http://files.innova-edu.webnode.ckeywords%20=%22b1%22,om/200003215-6a4f06b3b1/%20NormasUPEL2006.pdf>
- Huang, H., & Baddour, N. (2018). Bearing vibration data collected under time-varying rotational speed conditions [fecha de consulta: 24/6/2021]. *Data in Brief*, 21, 1745-1749. <https://doi.org/https://doi.org/10.1016/j.dib.2018.11.019>
- IBM. (s.f.-a). *Características y tipos de bases de datos* [Online; fecha de consulta 4/3/2022]. <https://developer.ibm.com/es/articles/tipos-bases-de-datos/>
- IBM. (s.f.-b). *Modelos estadísticos* [Online; fecha de consulta: 1/8/2021]. <https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=nodes-statistical-models>
- IBM. (s.f.-c). *What is MongoDB?* [Online; fecha de consulta 4/3/2022]. <https://www.ibm.com/cloud/learn/mongodb>

- Inc, G. (s.f.). *GitHub for enterprises* [Online; fecha de consulta 2/3/2022]. <https://github.com/enterprise>
- JavaTpoint. (s.f.). *Programming Language* [Online, blog; fecha de consulta 24/2/2022]. <https://www.javatpoint.com/programming-language>
- Kammermann, B., & Schwimmbeck, H. (2017). Reliability of induction machines: Statistics, tendencies, and perspectives [fecha de consulta: 15/4/2021], 1843-1847. <https://doi.org/10.1109/ISIE.2017.8001529>
- Koene, I., Klar, V., & Viitala, R. (2020). IoT connected device for vibration analysis and measurement [fecha de consulta: 26/4/2021]. *HardwareX*, 7, e00109. <https://doi.org/https://doi.org/10.1016/j.ohx.2020.e00109>
- Lacey, D. J. (s.f.). The Role of Vibration Monitoring in Predictive Maintenance [fecha de consulta: 14/4/2021]. https://www.schaeffler.com/remotemedien/media/_shared_media/08_media_library/01_publications/schaeffler_2/technicalpaper_1/download_1/the_role_of_vibration_monitoring.pdf
- LTD, N. E. (2021). *¿Qué es un Rodamiento?* [Online; fecha de consulta: 1/8/2021]. <https://www.nskeurope.es/es/bearings/products/what-s-a-bearing.html#>
- MDN. (2021). *What is the difference between webpage, website, web server, and search engine?* [Online; fecha de consulta: 2/8/2021]. https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Pages_sites_servers_and_search_engines
- Microsoft. (s.f.). *Database basics* [Online; fecha de consulta 4/3/2022]. <https://support.microsoft.com/en-us/office/database-basics-a849ac16-07c7-4a31-9948-3c8c94a7c204>
- MongoDB, I. (s.f.). *MongoDB Atlas Tutorial* [Online; fecha de consulta 4/3/2022]. <https://www.mongodb.com/basics/mongodb-atlas-tutorial>
- Mora, J. F. (2003). *Máquinas eléctricas* [Online; fecha de consulta: 20/6/2021]. McGraw-Hill.
- Mozilla & individual contributors. (s.f.-a). *About JavaScript What is JavaScript?* [Online; fecha de consulta 24/2/2022]. https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript
- Mozilla & individual contributors. (s.f.-b). *CSS: Cascading Style Sheets* [Online; fecha de consulta 24/2/2022]. <https://developer.mozilla.org/en-US/docs/Web/CSS>

- Mozilla & individual contributors. (s.f.-c). *HTML: HyperText Markup Language* [Online; fecha de consulta 24/2/2022]. <https://developer.mozilla.org/en-US/docs/Web/HTML>
- NumFOCUS. (s.f.). *pandas* [Online; fecha de consulta 27/2/2022]. <https://pandas.pydata.org/>
- NumPy. (s.f.). *NumPy* [Online; fecha de consulta 27/2/2022]. <https://numpy.org/>
- OpenCourseWare, M. (2016). *12. Testing Goodness of Fit (cont.)* [online; fecha de consulta 12/3/2022]. <https://www.youtube.com/watch?v=vMaKx9fmJHE>
- PAESSLER. (2021). *Cómo funciona un servidor* [Online; fecha de consulta: 1/8/2021]. <https://www.paessler.com/es/it-explained/server>
- Pinto, R., Rossetti, R., & Gonçalves, G. (2016). Wireless Sensor Network Simulation for Fault Detection in Industrial Processes [fecha de consulta: 26/4/2021], 333-338. <https://doi.org/10.5220/0006011003330338>
- Pressman, R. (2002). *INGENIERIA DE SOFTWARE* [Online; fecha de consulta: 24/7/2021]. McGraw-Hill Interamericana de España S.A.
- Ramírez, S. (s.f.). *FastApi* [Online; fecha de consulta 2/3/2022]. <https://fastapi.tiangolo.com>
- Sauer202 & col. (2021). Vibration — Wikipedia, The Free Encyclopedia [Online; fecha de consulta: 20/6/2021]. <http://en.wikipedia.org/w/index.php?title=Vibration&oldid=1034850880>
- SciPy. (s.f.). *SciPy* [Online; fecha de consulta 2/3/2022]. <https://scipy.org>
- sistema TDC. (s.f.). *¿Qué es un hosting? explicación completa* [Online; fecha de consulta 6/3/2022]. <https://sistematdc.com/blog/herramientas/que-es-un-hosting/>
- Smith, G. M. (2020). *Qué es Adquisición de Datos - DAQ o DAS?* [Online; fecha de consulta: 30/7/2021]. <https://dewesoft.com/es/daq/que-es-adquisicion-de-datos>
- Soto-Ocampo, C. R., Mera, J. M., Cano-Moreno, J. D., & Garcia-Bernardo, J. L. (2020). Low-Cost, High-Frequency, Data Acquisition System for Condition Monitoring of Rotating Machinery through Vibration Analysis-Case Study [fecha de consulta: 1/8/2021]. *Sensors*, 20(12). <https://doi.org/10.3390/s20123493>

- Themandrak & col. (2018). Web information system [Online; fecha de consulta: 31/7/2021]. https://en.wikipedia.org/wiki/Web_information_system
- Ugwiri, M., Mpia, I., & Lay-Ekuakille, A. (2020). Vibrations for fault detection in electric machines [fecha de consulta: 26/4/2021]. *IEEE Instrumentation and Measurement Magazine*, 23, 66-72. <https://doi.org/10.1109/MIM.2020.8979527>
- Vélez, C. M. (2005). *Apuntes de metodología de la investigación*. [Guía Universidad EAFIT, Medellín-Colombia, online; fecha de consulta 11/2/2022]. <https://www.docsity.com/es/apuntes-de-metodologia-de-la-investigacion-velez/6979855/>
- Vynith & col. (2019a). Dominio de la frecuencia [Online; fecha de consulta: 26/6/2021]. https://es.wikipedia.org/wiki/Dominio_de_la_frecuencia
- Vynith & col. (2019b). Dominio del tiempo [Online; fecha de consulta: 26/6/2021]. https://es.wikipedia.org/wiki/Dominio_del_tiempo
- Weber, M. (2012). Piezoelectric Accelerometers Theory and Application [Online; fecha de consulta: 20/6/2021].
- Wooley JC, e., Lin HS. (2005). *Catalyzing Inquiry at the Interface of Computing and Biology*. [Online; fecha de consulta: 30/7/2021]. National Academies Press (US).

BIBLIOGRAFÍA

- Adam Wathan, S. S. (2018). *Refactoring UI* [fecha de consulta: 5/11/2021].
- Alan Donovan, B. K. (2016). *The Go Programming Language* [fecha de consulta: 8/6/2021]. Addison-Wesley.
- GeeksforGeeks. (s.f.). *How to enable CORS headers in your Django Project?* [Online; fecha de consulta: 21/2/2022]. <https://www.geeksforgeeks.org/how-to-enable-cors-headers-in-your-django-project/>
- Google. (s.f.-a). *Documentation* [Online; fecha de consulta: 21/31/2022]. <https://go.dev/doc/>

- Google. (s.f.-b). *http2* [Online; fecha de consulta: 10/1/2022]. <https://pkg.go.dev/golang.org/x/net/http2>
- Posener, E. (2018). *HTTP/2 Adventure in the Go World* [Online; fecha de consulta: 12/1/2022]. <https://posener.github.io/http2/>
- Posener, E. (2019). *posener/h2conn* [Online; fecha de consulta: 12/1/2022]. <https://github.com/posener/h2conn>
- stackhawk. (s.f.). *Django CORS Guide* [Online; fecha de consulta: 21/2/2022]. <https://www.stackhawk.com/blog/django-cors-guide/>
- van der Veen, J. S., van der Waaij, B., & Meijer, R. J. (2012). Sensor Data Storage Performance: SQL or NoSQL, Physical or Virtual. *2012 IEEE Fifth International Conference on Cloud Computing*, 431-438. <https://doi.org/10.1109/CLOUD.2012.18>
- Wieruch, R. (2016). *the Road to learn React* [fecha de consulta: 8/5/2021]. Leanpub.
- Zakas, N. C. (2016). *Understanding ECMAScript 6* [fecha de consulta: 8/5/2021]. No starch press.

ANEXOS

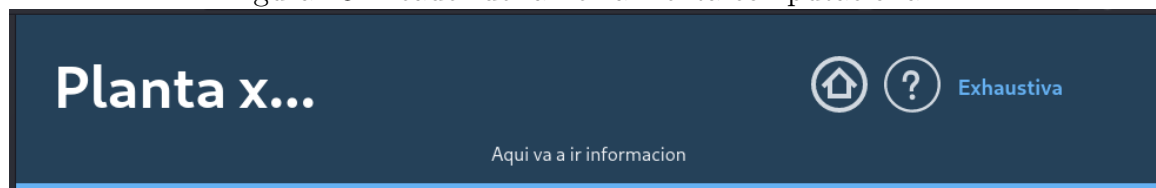
5.3. ANEXO A. Manual de Uso e Instalación

La herramienta computacional para el análisis de vibración en motores eléctricos facilita el conocimiento del estado actual de los motores en una planta al mismo tiempo que permite realizar estudios de evolución históricos y de vibración en frecuencia, permitiendo de este modo conocer en profundidad que motores podrían fallar y cual será el elemento, mecánico, que lo hará. Esto permite la realización de un mantenimiento predictivo, facilitando y mejorando la efectividad y coordinación de las paradas programadas en una empresa.

Esta herramienta es un sistema Web por lo cual **NO REQUIERE INSTALACIÓN EN LA COMPUTADORA**, solo se debe acceder al sitio Web (servidor) el cual se utilice como **HOST** para esta.

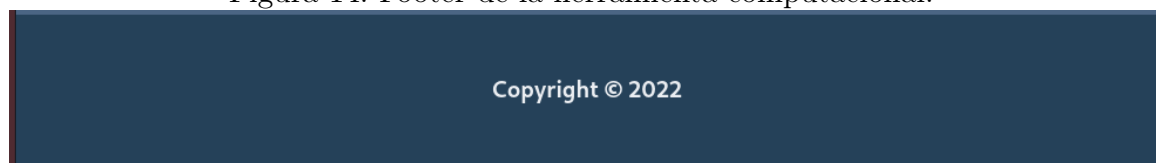
Para facilitar el acceso al usuario, esta consta de 3 paginas principales, general, especifica y exhaustiva, además consta de 2 elementos fijos que son el **Header**, mostrando la información de la planta y además permitiendo el regreso a la vista principal (general), solicitar ayuda y muestra en que vista se encuentra en el momento. Esto se puede observar en la figura 13.

Figura 13: Header de la herramienta computacional.



La otra vista constante es la del **Footer** que ofrece los derechos de autor y copyright.

Figura 14: Footer de la herramienta computacional.



El resto de la pantalla se modifica de acuerdo a que pestaña (vista) se observa, esto es:

5.3.1. Vista General

Es la vista principal de la aplicación, esta diseñada para un monitoreo constante lo que implica que recarga la información cada cierto tiempo (por defecto 30s) este periodo puede ser modificado únicamente por código y si se desea cambiar debe comunicarse con el proveedor de servicio.

Esta ventana, se observa en la figura 15 cuenta de dos partes fundamentales:

Figura 15: Vista de la pestaña general de la herramienta computacional.



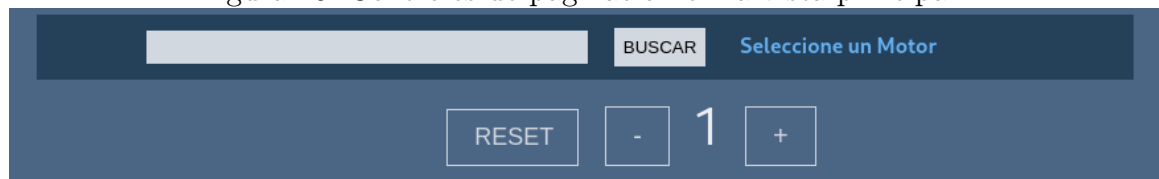
La primera es el despliegue de motores con un máximo de 12 motores por "pagina"(para modificar esta cantidad comunicarse con el proveedor de servicios) los cuales indican su código o identificador en su parte inferior (este código se ingresa al registrar el motor, se recomienda sea el mismo del de mantenimiento para facilitar su utilización) y un diagrama de motor con un fondo de color con código correspondiente a su estado, nivel de vibración y aceleración ajustado a los parámetros solicitados (modificaciones con el proveedor del servicio); este código es :

- verde: bien.
- amarillo: primera alerta.

- rojo: segunda alerta.

la segunda parte de interés es el buscador y control de paginación, se observa en la figura 16, el buscador permite solicitar la vista específica de cualquier motor, buscando por el Identificador del mismo; el control de paginación permite mostrar otros motores al "pasar" la página, de igual forma permite regresar directamente al principio con el botón de **RESET**.

Figura 16: Controles de paginación en la vista principal.



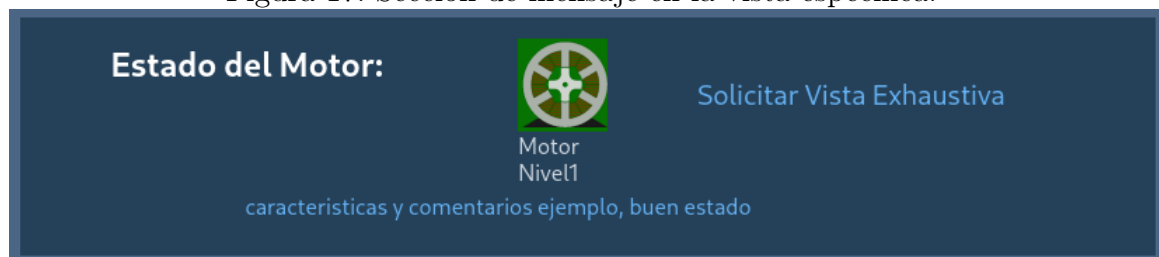
5.3.2. Vista Específica

Esta vista permite un nivel de estudio superior ya que es solamente enfocada a un motor y ofrece la información histórica del mismo, esto lo hace mediante gráficas y una tabla exportable a Excel.

tiene 3 secciones de trascendencia:

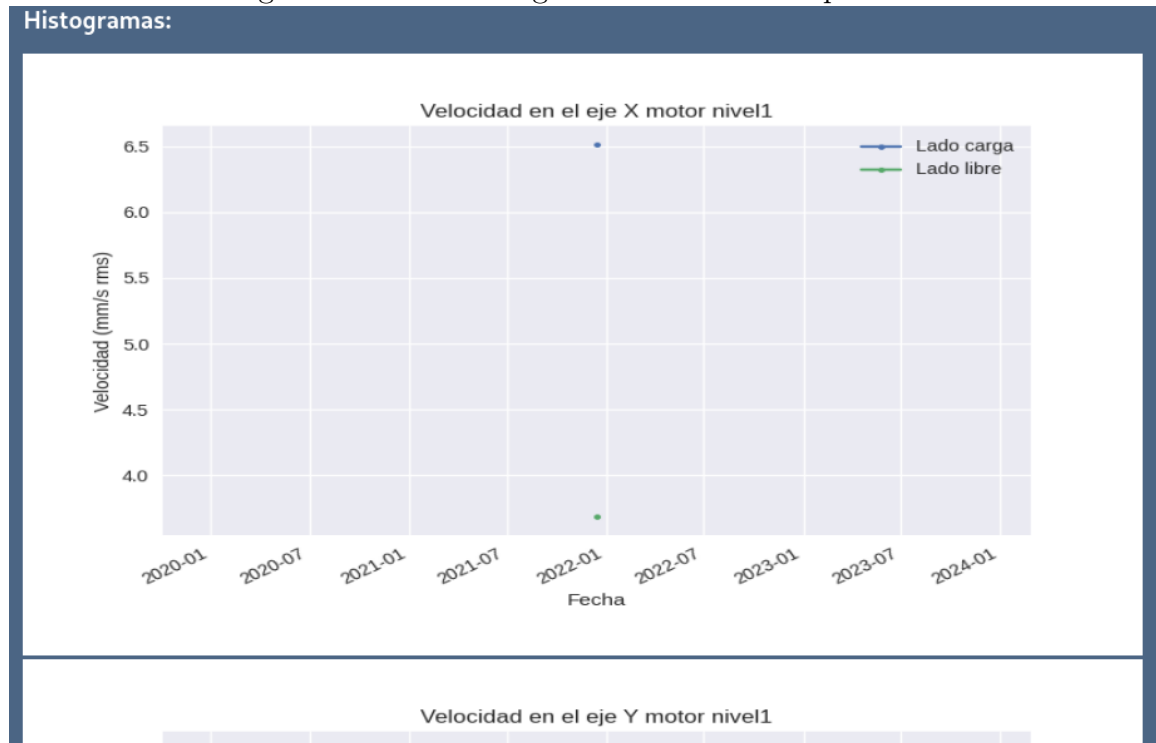
la primera es un encabezado resumen, se puede apreciar en 17; en este se encuentra el estado del motor como en la vista general, un enlace para solicitar la vista Exhaustiva y un mensaje con las características del motor o comentarios adicionales añadidos al momento de la incorporación del motor al sistema.

Figura 17: Sección de mensaje en la vista específica.



La segunda es una serie de gráficas de toda la información histórica, separada por ejes (x,y,z) y de la aceleración, se muestran después de la palabra **Histogramas**, como se observa en la figura 18.

Figura 18: Sección de gráficas en la vista específica.



La tercera es una tabla, como se observa en 19 exportable a Excel, mediante el botón, con toda la información de las mediciones a lo largo de la historia del sistema, esta posee los campos de:

- fecha, en la cual se tomo la medición.
- Id Sensor, identificador único, numero de serial, que posee el sensor (hardware), asociado.
- Lado, ubicación en la cual se tomo la medición, lado libre, lado con carga o chumaceras y acoples.
- Velocidad vertical, medición de la velocidad, en mm/s, medición rms
- Velocidad Horizontal, medición de la velocidad, en mm/s, medición rms

- Velocidad Axial, medición de la velocidad, en mm/s, medición rms
- Aceleración, medición de la aceleración, en g .

Nota, si la tabla no posee todos los campos en su navegador, selecciónela y utilice las flechas del teclado para redirigirla, o reduzca el tamaño de su pantalla.

Figura 19: Sección de la tabla en la vista específica.

Tabla resumen:

HISTOGRAMA COMO EXCEL


FECHA	ID SENSOR	LADO	VELOCIDAD VERTICAL	VELOCIDAD HORIZONTAL	VELOCIDAD AXIAL	ACELERACION
14/12/2021	15	Carga	5.330071390672369	6.514656414273091	3	0.33945888467322044
14/12/2021	7b	Libre	7.5652013532957	3.686068758829853	2	1.208568996642277

5.3.3. Vista Exhaustiva

Esta vista incluye las mismas características de la vista específica pero sufre una modificación en el encabezado resumen, como se puede apreciar en la figura 20 este redirige a la vista principal.

Figura 20: Sección de mensaje en la vista exhaustiva.

Estado del Motor:



Motor
Nivel1

características y comentarios ejemplo, buen estado

[Regresar A Vista General](#)

La otra modificación es que se añade otra sección de gráfica en la cual se muestra la salida de un estudio en frecuencia, permitiendo este su análisis; se observa en la figura

Figura 21: Sección de gráfica de frecuencia en la vista exhaustiva.

