Universidad Tecnológica de La Habana

José Antonio Echeverría



**Facultad de Ingeniería Informática**

**Desarrollo de tecnología de monitoreo y diagnóstico industrial.**

Informe de las Prácticas Profesionales 1

**Autor:** César Fernández García

**Tutores:** Dr. C. Juan C. Sepúlveda Peña

Ing. Juan Alejandro Baster Jiménez

**La Habana, Julio 2023**

# Resumen

En la actualidad Cuba está sufriendo las consecuencias de un embargo que afecta directa o indirectamente a todos los sectores sociales, incluyendo el industrial: donde específicamente se aprecia el deterioro de las maquinarias de generación de energía, compresores, etc. Estas maquinarias industriales necesitan de mantenimiento preventivo, para ello las herramientas de monitoreo y diagnóstico se encargan de mantener una vigilancia sobre los parámetros característicos de la máquina y son capaces de proveer la información necesaria para el diagnóstico de la misma.

A pesar de la existencia del embargo, el país compra las máquinas necesarias para la industria y aunque es considerablemente costoso, también los equipos para su monitoreo y control. Esto presenta una desventaja: cuando el equipo de monitoreo presenta una rotura irreparable, se pierde dinero necesario y la capacidad de monitoreo y control que estos proveen, afectando los procesos industriales.

Este trabajo tiene como objetivo desarrollar una aplicación multiplataforma de monitoreo y diagnóstico industrial. Los resultados de todo el proceso son, asimilación de la plataforma RAD Studio Versión 11.3. , asimilación de la teoría sobre análisis por vibraciones y concepto de la transformada rápida de Fourier, además de su análisis en el dominio del tiempo y frecuencia, logrando con esto un software multiplataforma que permite el monitoreo y diagnóstico de procesos industriales.

**Palabras claves**: sistemas de monitoreo y diagnóstico industrial, análisis de señales vibratorias, transformada rápida de Fourier.

# Abstract

Currently Cuba is suffering the consequences of an embargo that affects directly or indirectly all social sectors, including the industrial one: where specifically the deterioration of the machinery for generating energy, compressors, etc. is appreciated. These industrial machines need preventive maintenance, for this the monitoring and diagnostic tools are in charge of keeping a watch on the characteristic parameters of the machine and are capable of providing the necessary information for its diagnosis.

Despite the existence of the embargo, the country buys the necessary machines for the industry and although it is considerably expensive, also the equipment for its monitoring and control. This presents a disadvantage: when the monitoring equipment presents an irreparable breakdown, money is lost and the monitoring and control capacity that they provide, affecting the industrial processes. This work aims to develop a cross-platform application for industrial monitoring and diagnosis. The results of the whole process are, assimilation of the RAD Studio platform Version 11.3., assimilation of the theory on vibration analysis and concept of the fast Fourier transform, in addition to its analysis in the time and frequency domain, achieving with this a cross-platform software that allows monitoring and diagnosis of industrial processes.

**Keywords**: industrial monitoring and diagnostic systems, vibration signal analysis, fast Fourier transform.

**Anexo A. Plan individual de tareas del estudiante**

|  |  |  |
| --- | --- | --- |
| **Tareas** | **Fecha de entrega** | **Rol(es) que desarrolla(n) con la tarea** |
| Reunión de inicio de la práctica | 12/6/2023 | - |
| Asimilación de la plataforma RAD Studio Versión 11.3 | 20/6/2023 | PG |
| Asimilación de la teoría sobre análisis por vibraciones y concepto de la (Transformada rápida de Fourier) FFT. Análisis en el dominio del tiempo y de la frecuencia. | 20/6/2023 | AS |
| Análisis y diseño de una plataforma escalable de captura y procesamiento de datos. | 23/6/2023 | AR |
| Programar una aplicación Multiplataforma (Windows, Android) que sea capaz de seleccionar el *driver* de captura de datos (.dll en Windows, y Android) y usarla para capturar datos, mostrar dichos datos gráficamente en el dominio del tiempo y de la frecuencia (usando la (Transformada rápida de Fourier) FFT) y almacenarlos en una base de datos para futuro post-procesamientos. | 30/6/23 | PG |
| Entrega primera versión del software para revisión | 30/6/23 |  |
| Entrega versión final del software | 15/7/23 |  |
| Elaborar informe de la práctica | 5/7/23 | EE |
| Entregar informe de la práctica al tutor | 5/7/23 | EE |
| Rectificar señalamientos del informe | 10/7/23 | EE |
| Entrega del informe final de la práctica | 17/7/23 | EE |
| Defensa de la práctica | 19-21/7/23 | Todos |

**Índice**

[Resumen 1](#_Toc140490306)

[Abstract 2](#_Toc140490307)

[Introducción 1](#_Toc140490308)

[Capítulo 1: Fundamentación teórica. 4](#_Toc140490309)

[1.1 Estado del arte. 4](#_Toc140490310)

[1.2 Análisis por vibraciones de procesos industriales. 6](#_Toc140490311)

[1.3 Monitoreo. 8](#_Toc140490312)

[1.4 Sistemas de monitoreo. 8](#_Toc140490313)

[1.5 Transformada rápida de Fourier (FFT). 9](#_Toc140490314)

[1.6 Mantenimiento predictivo. 11](#_Toc140490315)

[1.7 Análisis de las tecnologías. 12](#_Toc140490316)

[1.7.1 Lenguaje de programación. 12](#_Toc140490317)

[1.7.2 Entorno de desarrollo integrado (IDE). 13](#_Toc140490318)

[1.7.3 Motor de base de datos. 14](#_Toc140490319)

[Conclusiones parciales 15](#_Toc140490320)

[Capítulo 2: Solución propuesta 16](#_Toc140490321)

[2.1 Explicación de la solución. 16](#_Toc140490322)

[2.2 Diseño de la base de datos. 16](#_Toc140490323)

[2.2.1 Modelo lógico de los datos. 16](#_Toc140490324)

[2.2.2 Modelo físico de los datos. 18](#_Toc140490325)

[2.3 Diagramas de actividades. 19](#_Toc140490326)

[2.4 Seguridad, roles y privilegios. 23](#_Toc140490327)

[2.5 Diseño de la interfaz. 24](#_Toc140490328)

[2.6 Tratamiento de errores. 28](#_Toc140490329)

[Conclusiones parciales. 30](#_Toc140490330)

[Capítulo 3: Validación de la solución 32](#_Toc140490331)

[3.1 Pruebas de caja negra. 32](#_Toc140490332)

[3.1.1 Insertar usuario HU 1. 32](#_Toc140490333)

[3.1.2 Editar usuario HU 2. 35](#_Toc140490334)

[3.1.3 Eliminar usuario HU 3. 38](#_Toc140490335)

[3.1.4 Insertar ruta HU 4. 41](#_Toc140490336)

[3.1.5 Insertar máquina HU 5. 43](#_Toc140490337)

[3.1.6 Insertar señal HU 6. 46](#_Toc140490338)

[3.1.7 Autenticar HU 7. 48](#_Toc140490339)

[Conclusiones parciales. 51](#_Toc140490340)

[Conclusiones 52](#_Toc140490341)

[Recomendaciones 53](#_Toc140490342)

[Referencias bibliográficas 54](#_Toc140490343)

# Introducción

La Industria 4.0 es la materialización de la transformación digital del sector industrial, que ofrece toma de decisiones en tiempo real y una productividad, mantenibilidad, flexibilidad y agilidad mayores. La Industria 4.0 se caracteriza por la creciente automatización y el uso de máquinas y fábricas inteligentes que se integran con tecnologías como el internet de las cosas (IoT, por sus siglas en inglés), análisis y cloud computing, IA y machine learning. La Industria 4.0 también implica el desarrollo de fábricas inteligentes que proporcionan una oportunidad increíble a la industria: les abre el camino a la cuarta revolución industrial (¿Qué es la Industria 4.0 y cómo funciona? | IBM, 2023).

La gran crisis económica mundial que se vive en la actualidad afecta a todos los aspectos de la sociedad, lo que se refleja en el deterioro de las maquinarias de transporte y de la tecnología en general. La obsolescencia de las maquinarias relacionadas con la producción de cualquier índole conlleva al desgaste de las piezas que las componen, lo que a la larga provoca roturas irreparables y afecta a los diferentes procesos de producción. Además, es muy difícil obtener nuevas máquinas y piezas debido al bloqueo económico-financiero impuesto por los Estados Unidos. Una de las causas del deterioro y rotura de las máquinas industriales es la falta de mantenimiento y reemplazo de las piezas en el momento adecuado. Debido a las causas antes mencionadas, estos elementos no son suficientes para todas las maquinarias que intervienen en los diferentes procesos de producción. Por lo tanto, es necesario detectar a tiempo aquellos equipos que necesitan más intervención preventiva. Esto es posible si se pueden detectar los indicios que provocan el desgaste y las roturas antes mencionadas, lo cual se puede lograr con el uso de un método capaz de monitorear los parámetros que caracterizan las vibraciones de dichas maquinarias. Sin embargo, el problema radica en que se carece de este método.

En la actualidad han surgido máquinas que utilizan componentes rotatorios que generan vibraciones, tales como; compresores de alta velocidad, turbinas de vapor y gas, generadores, bombas, etc. Aunque por lo general son resistentes y bien diseñados, los ejes en operación son muchas veces vulnerables a defectos que se desarrollan de manera imprevista. Dada la problemática de no poder detectar de forma prematura estas vibraciones (imperceptibles o no) tan peligrosas, en los equipos ocurren averías o roturas, ejemplo de estas son desalineamiento, desequilibrio, desgaste, y, por transitividad el paro de la producción (Zambrano-Reyes, Nossov, & Gómez-Mancilla) (Mohd Ghazali & Rahiman, 2021).

Esta problemática se ha intentado solucionar en el país de una manera a veces inadecuada, ya que estas soluciones no toman en cuenta las restricciones internacionales que presenta el país debido al embargo económico, por ejemplo: falta de presupuesto, falta de piezas de repuesto, la compra de piezas y máquinas a terceros países por un precio mayor, demora en la obtención de estos elementos y complementos. (Ministerio de Relaciones Exteriores de Cuba, 2023)

Teniendo en cuenta lo anterior descrito se puede identificar como problema:

¿Cómo desarrollar un software para el monitoreo y diagnóstico en tiempo real de máquinas rotatorias industriales?

El objeto de estudio está enmarcado en los diferentes aplicaciones existentes que establezcan el monitoreo sobre máquinas generadoras de vibraciones, así como la seguridad y almacenamiento de los datos procesados de señales y del software en sí, y que permitan un análisis de tendencia de los datos procesados, teniendo como campo de acción las aplicaciones de escritorio y móviles que permitan, el procesamiento de señales generadas por sensores de vibraciones en máquinas industriales.

Como objetivo general se plantea desarrollarun software multiplataforma para el monitoreo y diagnóstico industrial.

A partir de lo previamente mencionado se trazaron los siguientes objetivos específicos:

1. Investigar sobre otras aplicaciones que realicen el monitoreo y diagnóstico de procesos industriales.
2. Diseñar un software multiplataforma para el monitoreo de las vibraciones en máquinas rotatorias que realizan algún proceso industrial.
3. Implementar funcionalidad que permita la visualización en tiempo real de las señales capturadas tanto en el dominio del tiempo como en el de la frecuencia.
4. Implementar funcionalidad para almacenar en una base de datos las señales y los resultados provenientes de procesar cada señal.
5. Implementar mecanismos de seguridad que aseguren que el software no sea utilizado por personal no autorizado.
6. Validar la solución propuesta.

Para el cumplimiento de dichos objetivos se definieron las siguientes tareas:

1. Investigar y analizar sobre aplicaciones que permitan el monitoreo, visualización de señales, almacenamiento de su procesamiento y análisis de tendencia de esos procesamientos.
2. Asimilar la teoría sobre análisis por vibraciones y concepto de la transformada rápida de Fourier (FFT, por sus siglas en inglés).
3. Asimilar el lenguaje de programación Pascal, así como de la plataforma RAD Studio Versión 11.3 y el sistema gestor de bases de datos SQLite.
4. Analizar y diseñar una plataforma escalable de captura y procesamiento de datos.
5. Programar una aplicación multiplataforma que sea capaz de seleccionar el *driver* de captura de datos y usarla para capturar datos, mostrar dichos datos gráficamente en el dominio del tiempo y de la frecuencia, y almacenarlos en una base de datos para futuro post-procesamientos.
6. Realizar pruebas de software de caja negra.

**Valor práctico**

El valor práctico de este trabajo reside en el desarrollo de una aplicación multiplataforma que permita al usuario el monitoreo y almacenamiento del procesamiento de señales proveniente de máquinas industriales de cualquier tipo que produzcan vibraciones, permitiendo así, un diagnóstico más rápido y de mayor calidad por parte del usuario de las maquinarias involucradas.

# Capítulo 1: Fundamentación teórica.

En este capítulo se presentan los principios teóricos fundamentales para comprender adecuadamente el trabajo realizado. Se abordan temas relacionados con el estado del arte de las aplicaciones para el monitoreo de máquinas industriales y el análisis de vibraciones en procesos industriales, específicamente en máquinas con ejes rotatorios. Además, se explican conceptos como el monitoreo y los sistemas de monitoreo. También se discute la Transformada Rápida de Fourier (FFT) y el mantenimiento predictivo. Finalmente, se hace un estudio y posterior selección de las tecnologías utilizadas para la confección de la propuesta.

## 1.1 Estado del arte.

Existen varias aplicaciones que permiten el monitoreo de señales provenientes de máquinas industriales y que también permiten el análisis espectral de las señales. Algunos ejemplos incluyen **DynaPredict** (Dynamox, 2023), **BK Connect** (Brüel & Kjær, 2023) y **PRTG** (PAESSLER, 2023). Estos software ofrecen soluciones para el monitoreo del estado de las máquinas y los componentes, y pueden ser utilizados para el mantenimiento predictivo.

Acerca de DynaPredict:

DynaPredict es una solución de monitoreo de la condición de las máquinas desarrollada por Dynamox. Esta solución se basa en el uso de un DynaLogger, un data logger Bluetooth con sensores de vibración y temperatura para supervisar el estado de la máquina y realizar un análisis espectral triaxial. Los parámetros medidos se muestran instantáneamente en el smartphone y su historial de datos se almacena en la plataforma web para su análisis y toma de decisiones. (Dynamox, 2023)

DynaPredict permite a los usuarios supervisar el estado de sus máquinas en tiempo real y tomar medidas preventivas para evitar fallas y tiempos de inactividad. La recolección de datos del DynaLogger está automatizada por una pasarela, el DynaGateway, desarrollada por Dynamox. Esto permite una supervisión continua y sin interrupciones del estado de las máquinas. (Dynamox, 2023)

Acerca de BK Connect:

BK Connect es un software de análisis de sonido y vibraciones creado por la compañía Brüel & Kjær. Este software ofrece una amplia variedad de aplicaciones y opciones diseñadas para tareas generales de sonido y vibraciones. Constituye un conjunto de herramientas flexible, capaz de hacer grabaciones con un solo clic o de resolver problemas complejos, que permiten desde gestionar datos hasta elaborar informes (Brüel & Kjær, 2023).

Las aplicaciones esenciales de BK Connect tienen una estructura modular, orientada a los roles de los distintos usuarios, y forman la base de un sistema flexible de medida de sonido y vibraciones. Cada aplicación puede funcionar como un conjunto de herramientas autónomo o combinarse con otras aplicaciones. Es posible combinar las aplicaciones esenciales y sus opciones con otra funcionalidad avanzada, para crear soluciones optimizadas, a la medida de su flujo de trabajo (Brüel & Kjær, 2023).

Además, BK Connect ofrece applets diseñados para realizar tareas de ensayo muy específicas. Cada applet individual es un módulo de software autónomo, adaptado a las necesidades del usuario y concebido para realizar una tarea específica (Brüel & Kjær, 2023).

Acerca de PRTG:

PRTG es un software de monitoreo que incorpora elementos de todas las áreas, para que pueda supervisar la salud, el estado y la condición de máquinas, sistemas de control, dispositivos y más en entornos de Tecnología de la Información (TI, por sus siglas en inglés) y Tecnología de Operaciones (OT, por sus siglas en inglés). Este software es compatible de fábrica con estándares y protocolos industriales comunes como OPC UA, MQTT, Modbus TCP y más. Además, permite el monitoreo de dispositivos Ethernet industriales en su entorno de OT y la visualización de su entorno en paneles que incluyen elementos de TI, OT e Internet de las cosas industrial (IIoT, por sus siglas en inglés) (PAESSLER, 2023).

PRTG también ofrece la posibilidad de extender la funcionalidad de las soluciones comunes de la industria para proporcionar una funcionalidad ampliada. Proporciona sensores nativos para Modbus TCP y RTU y permite supervisar si los clientes pueden conectarse y suscribirse a su bróker de MQTT y publicar a través de él con el sensor ida y vuelta MQTT (PAESSLER, 2023).

## 1.2 Análisis por vibraciones de procesos industriales.

El diagnóstico de maquinarias por vibraciones, como una de las tantas técnicas involucradas en el mantenimiento predictivo, se basa en el monitorizado de las vibraciones producidas por las maquinarias en funcionamiento y, debido a las conveniencias de su aplicación, constituye uno de los más preferidos por los especialistas del mantenimiento. Las técnicas de análisis de vibraciones para el diagnóstico que más se aplican son extraídas fundamentalmente del campo del procesamiento de señales (Mohd Ghazali & Rahiman, 2021).

El análisis por vibraciones es una técnica que consiste en medir y analizar las oscilaciones o movimientos alternos de ciertos puntos de una máquina o estructura, para determinar su estado o condición, así como la fuente y la gravedad de posibles fallas. El análisis por vibraciones puede ayudar a prevenir y predecir problemas como desbalance, desgaste, desalineación, defectos en rodamientos, fricción, grietas en engranajes, etc. (Mohd Ghazali & Rahiman, 2021)

Para realizar el análisis por vibraciones se utilizan instrumentos como sensores, analizadores y colectores de datos, que captan y procesan las señales de vibración en el dominio del tiempo o de la frecuencia. Estas señales pueden ser representadas como formas de onda, espectros o diagramas que contienen información útil para el diagnóstico. Existen diferentes métodos y técnicas para extraer y clasificar las características de las señales de vibración, como el análisis estadístico, la transformada rápida de Fourier (FFT), el análisis de envolvente, el análisis espectral, la transformada wavelet (WT, por sus siglas en inglés), el análisis cepstral, la transformada de Hilbert-Huang (HHT, por sus siglas en inglés), entre otros. Cada método tiene sus ventajas y desventajas, y se aplica según el tipo de máquina, la frecuencia de vibración y el tipo de falla que se quiere detectar (Mohd Ghazali & Rahiman, 2021).

En el mantenimiento predictivo, el análisis por vibraciones es una técnica que permite supervisar y diagnosticar la maquinaria rotativa para predecir posibles fallas y planificar el mantenimiento preventivo.

Como resultado del análisis de una vibración en el dominio del tiempo, a partir de la cual se calculan parámetros como RMS o Valor efectivo, PICO y PICO-PICO. De ellos el más utilizado es el RMS que está asociado a la potencia de la vibración y este se determina de forma discreta de la siguiente forma: (Marín, 1997)

(1.1)

El valor PICO es el valor máximo de una magnitud (aceleración, velocidad, desplazamiento) que varía en un intervalo de tiempo (Marín, 1997). El valor PICO-PICO (de un evento oscilatorio) es la diferencia algebraica entre los valores extremos de una magnitud que varía en un intervalo de tiempo (Marín, 1997). En la [Figura 1](#Figura1) se observa la representación de los parámetros mencionados anteriormente en el dominio del tiempo (Marín, 1997)

Figura 1 Parámetros característicos de una señal.

La detección del valor PICO-PICO se utiliza para mediciones de desplazamiento y los de PICO y RMS se usan para mediciones de velocidad y aceleración (Marín, 1997).

Los resultados del análisis por vibraciones se utilizan en el mantenimiento predictivo para monitorear el rendimiento y la condición de las máquinas y predecir posibles fallas. Al medir y analizar regularmente las señales de vibración de las máquinas, se pueden detectar cambios en su comportamiento que indiquen la presencia de una falla o un problema potencial. Esto permite a los técnicos de mantenimiento tomar medidas preventivas, como ajustar, reparar o reemplazar componentes antes de que fallen y causen daños mayores o interrupciones en la producción.

## 1.3 Monitoreo.

Monitoreo viene de la palabra monitor, es decir, una pantalla por la cual se pueden ver imágenes en tiempo real (Ferrer, 2023).

El monitoreo es el acto de supervisar los cambios en el estado y el flujo de datos en un sistema. Hay dos formas de hacerlo: proactiva y reactiva. La forma proactiva implica observar indicadores visuales, como series de tiempo y tableros de información, lo que se conoce como monitoreo. La forma reactiva implica enviar notificaciones automáticas a los operadores para informarles sobre cambios en el sistema, lo que se conoce como alerta (de Murga Aguiar, 2020).

Las alertas son la capacidad de un sistema de monitoreo para detectar y notificar a los operadores sobre un evento importante que provoca un cambio significativo en el estado. Estas notificaciones, conocidas como alertas, pueden adoptar diversas formas, como correo electrónico, SMS, mensajes instantáneos o llamadas telefónicas (de Murga Aguiar, 2020).

En general, el monitoreo consiste en la observación del curso de uno o más parámetros para detectar eventuales anomalías.

## 1.4 Sistemas de monitoreo.

Un sistema de monitoreo es responsable de hacer un seguimiento del estado del sistema completo, incluyendo la infraestructura y otros subsistemas, para asegurar la fiabilidad y estabilidad de los servicios que provee. Consta de una serie de componentes de software que realizan mediciones y recolectan, almacenan e interpretan los datos monitoreados. El sistema está optimizado para almacenar de manera eficiente y para generar métricas de monitoreo, que posteriormente son mostradas en series de tiempo y cuyos puntos de datos son analizados con la finalidad de emitir alertas. Observa y sigue en el tiempo las operaciones y actividades de los usuarios, aplicaciones y servicios de red, registrando todas las acciones y alertando si identifica alguna violación o brecha que resulte en un comportamiento anormal (de Murga Aguiar, 2020).

La mayoría de los sistemas de monitoreo tienen una estructura parecida. El procedimiento inicia con la recepción de datos, después el cliente recolecta y transmite los datos al sistema de monitoreo mediante una interfaz especializada. El sistema guarda estos datos en métricas y remite los nuevos puntos de datos para valorar las condiciones de los umbrales. Cuando se percibe que un umbral ha sido transgredido, se manda una notificación al operador acerca del fallo. La [Figura 2](#Figura2) ilustra las interacciones entre los componentes de un sistema de monitoreo (de Murga Aguiar, 2020).



Figura 2 Interacciones entre los componentes de la arquitectura de los sistemas de monitoreo.

## 1.5 Transformada rápida de Fourier (FFT).

Las funciones de análisis se pueden calcular mediante las integrales que la definen, pero el núcleo principal del análisis de señales y del análisis de sistemas se realiza por el análisis por Fourier sustentadas por las siguientes ecuaciones conocidas como transformada discreta de Fourier (DFT) (Marín, 1997):

(1.1)

(1.2)

Para n=0, 1,…, N-1; k=0, 1,…, N-1.

Siendo N el número de muestras discretas de un segmento de las señal vibroacústica. En el caso de las vibraciones como señales a procesar, y(n) es real y Y(k) es compleja. Al aplicar la transformada discreta de Fourier a un número N de muestras se muestra en la [Figura 3](#Figura3) (Marín, 1997).



Figura 3 Representación compleja de la transformada discreta de Fourier (DFT).

Hasta el año 1965 aproximadamente se usó la DFT en computadoras con programas que tenían que ejecutar operaciones de cálculo, siendo N el número de muestras discretas en un bloque de datos. Para ese mismo año se da a conocer la FFT, popularizada por James Willian Cooley de IBM y John W. Turkey de Bell Laboratories requiriendo solamente operaciones para transformar un bloque de N datos por mucha diferencia mejor que la DFT (Marín, 1997) (Davuluru, Hettiarachchi, & Balster, 2022).

En resumen la FFT es un algoritmo eficiente para calcular la DFT y su inversa. La DFT es una herramienta matemática para el análisis de señales en el dominio de la frecuencia, pero su cálculo directo puede ser muy costoso en términos de tiempo y recursos computacionales. La FFT reduce significativamente la complejidad computacional del cálculo de la DFT, lo que la hace más adecuada para aplicaciones en tiempo real.

## 1.6 Mantenimiento predictivo.

Para entender el concepto de mantenimiento predictivo, se puede consultar el diccionario Oxford, donde el término "Predictivo" se define como "relacionado con la capacidad de prever el futuro", mientras que "mantenimiento" se refiere a "la acción de mantener algo en buen estado mediante revisiones o reparaciones periódicas". Al combinar estos dos conceptos, podemos definir el mantenimiento predictivo como "la acción y capacidad de mantener algo en buenas condiciones para prever lo que sucederá en el futuro" (Ramos Cosio, 2020).

El uso del mantenimiento predictivo ofrece varias ventajas, como (Ramos Cosio, 2020):

1. Reducción de los tiempos de parada.
2. Prevención del funcionamiento de maquinarias en condiciones de riesgo.
3. Disminución de la interrupción del servicio en otros equipos en funcionamiento, evitando distorsiones en otras áreas.
4. Optimización de la gestión del área de mantenimiento.
5. Verificación y diagnóstico basados en los resultados de las mediciones del estado actual del equipo.
6. Determinación precisa del tiempo máximo de operación sin riesgo de fallas imprevistas.
7. Facilitación de la toma de decisiones sobre la parada de algún equipo en funcionamiento.
8. Ahorro significativo en tiempo y dinero al evitar paradas por mantenimiento preventivo en instalaciones en funcionamiento.
9. Aumento de la confiabilidad en equipos, materiales e instalaciones en general.

En resumen el mantenimiento predictivo da la capacidad de realizar mantenimiento donde puede haber un problema o falla en el futuro, para así obtener ventajas y evitar problemas de diferente índole.

## 1.7 Análisis de las tecnologías.

Después de decidir que se necesita crear un nuevo software o añadir una nueva funcionalidad, es esencial realizar un examen detallado de las tendencias y una investigación del estado actual de las distintas tecnologías relacionadas con el desarrollo de la propuesta. Este análisis debe ser exhaustivo, cubriendo todos los aspectos a considerar de componentes y tecnologías adecuadas para las tareas que demanda la aplicación.

### 1.7.1 Lenguaje de programación.

Pascal es un lenguaje de programación creado por el profesor suizo Niklaus Wirth entre los años 1968 y 1969, y publicado en 1970. Su objetivo era crear un lenguaje que facilitara el aprendizaje de programación a sus alumnos, utilizando la programación estructurada y estructuración de datos (Cantú, 2003) (Wikipedia, 2023).

Algunas características de Pascal son (Wikipedia, 2023) (Cantú, 2003):

1. Es un lenguaje de programación fuertemente tipado, lo que significa que el tipo de dato de todas las variables debe ser declarado previamente para que su uso quede habilitado.
2. El código está dividido en porciones fácilmente legibles llamadas funciones o procedimientos, lo que facilita la utilización de la programación estructurada en oposición al antiguo estilo de la programación monolítica.
3. Utiliza el símbolo `:=` para la asignación en vez de `=`, lo que ayuda a prevenir errores comunes donde se utiliza el símbolo de igualdad para comparar valores en lugar del comparador `==`.
4. Sus programas tienen definidas dos partes: declarativa y ejecutiva. En la primera debe aparecer todo lo que se usará en la segunda, lo que ayuda a prevenir errores y mejorar la legibilidad del código.

Algunas de las ventajas de Pascal son (Lenguaje de programación Pascal actualizado a 2023, 2023):

1. Promueve un método disciplinado y elegante a la hora de programar, con programas bien organizados, claros y relativamente libres de errores.
2. Es un lenguaje orientado para cualquier tipo de ordenador, ya sea un gran sistema de cómputo o una simple computadora personal de sobremesa.
3. Es un lenguaje que casi se considera pseudo-código, lo que facilita la comprensión y explicación del código.
4. Fomenta los buenos principios de una buena práctica de programación, como la integridad, la simplicidad, la modularidad y la generalidad.

Para la solución del objetivo general se va a usar la evolución de Pascal, llamada Object Pascal, una versión que ofrece las facilidades del paradigma de la programación orientada a objeto y además que la solución del proyecto tiene antecedentes de código ya escrito y se va a poder reutilizar.

### 1.7.2 Entorno de desarrollo integrado (IDE).

Para la creación de la aplicación se va a usar el IDE Embarcadero Delphi 11 versión 28.0.47991.2819, algunas características importantes son (Delphi Características, 2023) (Delphi 11.3, 2023):

1. Esta versión de Delphi incluye soporte para Markdown y un simulador de iOS en macOS ARM 64, lo que permite a los desarrolladores utilizar el lenguaje de marcado Markdown y probar aplicaciones para iOS en dispositivos macOS con procesadores ARM 64.
2. También migra el depurador de Delphi Linux a LLDB, un depurador de código abierto que ofrece características avanzadas y puede mejorar la experiencia de depuración para los desarrolladores que utilizan Delphi para crear aplicaciones para Linux.
3. Además, esta versión incluye mejoras en la usabilidad, el rendimiento y la estabilidad en varias áreas, como el IDE, los compiladores, las cadenas de herramientas, la UX, la RTL, la base de datos y las bibliotecas de Internet.
4. Permite la construcción rápida de aplicaciones multiplataforma con una base de código común para Windows, Android, iOS, macOS y Linux.
5. Ofrece Object Pascal moderno con bibliotecas de componentes y compiladores nativos para múltiples plataformas.

En resumen significa que esta versión de Delphi presenta la posibilidad de trabajo con el motor base de datos SQLite, permite que la aplicación sea multiplataforma y llegue a la mayor cantidad posible de usuarios finales. Además, Delphi proporciona una gran cantidad de componentes visuales y no visuales disponible para el desarrollo de aplicaciones.

### 1.7.3 Motor de base de datos.

Se ha elegido a SQLite debido a que, este es una biblioteca escrita en lenguaje C que implementa un motor de base de datos SQL pequeño, rápido, autónomo, confiable y con muchas funciones. Además, SQLite es el motor de base de datos SQL más implementado en el mundo y se utiliza ampliamente en todos los teléfonos inteligentes, dispositivos de Internet de las cosas, además de venir incluido en innumerables otras aplicaciones que las personas usan todos los días. La versión a usar es la más reciente de SQLite en el momento la 3.42.0. (SQLite, What Is SQLite?, 2023) (Team).

SQLite puede ejecutarse en múltiples plataformas. Almacena toda la base de datos (definiciones, tablas, índices y los datos en sí) como un solo archivo multiplataforma en una máquina host. El formato de archivo de la base de datos es multiplataforma, lo que significa que puedes copiar libremente una base de datos entre sistemas de 32 y 64 bits o entre arquitecturas big-endian y little-endian. Esto hace que SQLite sea una opción conveniente para el desarrollo de aplicaciones multiplataforma (SQLite, Single-file Cross-platform Database, 2023) (SQLite, About SQLite, 2023) (SQLite, System Requirements For SQLite, 2023).

SQLite no es directamente comparable con motores de bases de datos SQL cliente/servidor como MySQL, Oracle, PostgreSQL o SQL Server, ya que SQLite intenta resolver un problema diferente. Los motores de bases de datos SQL cliente/servidor se esfuerzan por implementar un repositorio compartido de datos empresariales. Hacen hincapié en la escalabilidad, la concurrencia, la centralización y el control. SQLite, por otro lado, se esfuerza por proporcionar almacenamiento de datos local para aplicaciones y dispositivos individuales. SQLite hace hincapié en la economía, la eficiencia, la fiabilidad, la independencia y la simplicidad. SQLite no compite con las bases de datos cliente/servidor (SQLite, Appropriate Uses For SQLite, 2023).

En conclusión se puede afirmar que SQLite cumple las características de ser rápido, autónomo, eficiente, compacto y permite que la gestión de bases de datos en múltiples plataformas.

## Conclusiones parciales

Con la culminación de este capítulo se puede arribar a las siguientes conclusiones:

1. Los resultados obtenidos en el análisis por vibraciones al ser usados en el mantenimiento predictivo previene fallas en las maquinarias industriales.
2. Al realizar el análisis por vibraciones mediante la FFT se realizan menos operaciones de cálculo por parte del software, trayendo consigo mayor velocidad.
3. Los sistemas de monitoreo son utilizados para la compilación de datos, su procesamiento y presentación de los mismos.
4. Utilizar las tecnologías presentadas hacen que se cumplan los requisitos que tiene que cumplir la aplicación con eficiencia, rapidez y compatibilidad.

# Capítulo 2: Solución propuesta

En este capítulo se aborda el diseño que da solución al problema antes mencionado en la introducción. Para ello se hace uso de diagramas UML y se realiza el diseño de la base de datos.

## 2.1 Explicación de la solución.

Se desarrolló un software que permite el monitoreo y diagnóstico industrial. Para contribuir al aspecto del monitoreo, se creó una interfaz que visualiza tanto la señal vibratoria como el espectro de la misma. Para la visualización de la señal vibratoria como del espectro, se utiliza un driver que realiza la conversión analógico-digital de la señal y halla su espectro mediante el cálculo de la transformada rápida de Fourier. Para mostrar al usuario los gráficos se utiliza el componente visual TChart.

En el diagnóstico de los equipos industriales participa principalmente el usuario o especialista que conoce las características del proceso industrial, y el software contribuye a esto con un apartado de análisis de tendencia, que muestra, tras un cálculo histórico de los parámetros característicos de cada señal registrada en la base de datos, los valores de tendencia central, tales como media aritmética, media y mediana, etc.

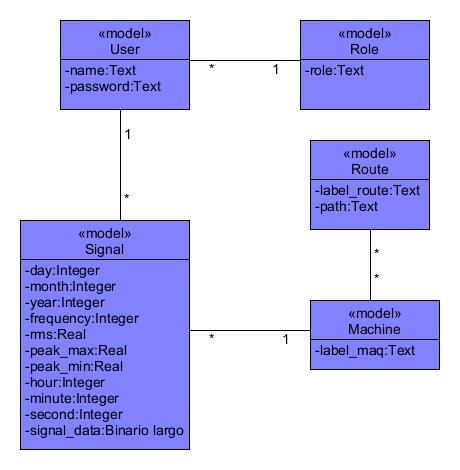
Para el cumplimiento de la gestión de los datos, se crearon apartados para la gestión de usuarios, rutas, señales y máquinas. Los cuales cumplen con su objetivo principal de forma rápida y eficaz.

## 2.2 Diseño de la base de datos.

Los modelos de datos sirven para diseñar la estructura de los almacenes de datos persistentes que se utilizan en un sistema (Matos, 2010). A continuación, se muestran los resultados del modelado lógico y físico.

### 2.2.1 Modelo lógico de los datos.

En el siguiente modelo lógico de datos, se identifican las entidades clave y las relaciones que capturan la información primordial que la aplicación necesita persistir en la base de datos.



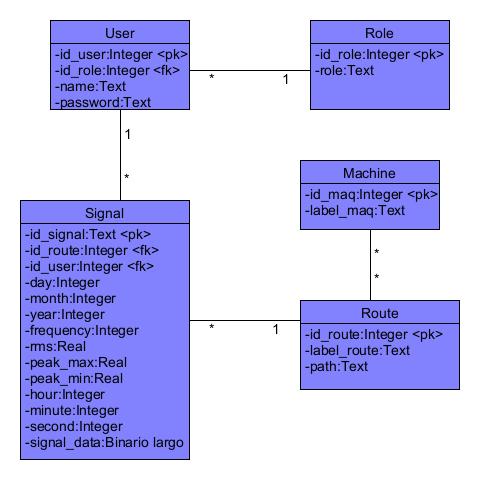
**Figura 4 Modelo lógico de los datos**

Consideraciones hechas al modelo lógico de los datos:

* Signal se registra como campos el day, month, year, hour, minute, second, ya que el motor de bases de datos SQLite no soporta el tipo de dato fecha.
* Signal se registra como campos calculables rms, peak\_max, peak\_min a partir del campo signal\_data, pero a pesar de esto se tomó la decisión de almacenarlo, ya que estos se calculan en tiempo real mediante la captura de la señal.
* Machine solo existe como etiquetador para facilidades del negocio.

### 2.2.2 Modelo físico de los datos.

Como resultado del análisis y adecuación del modelo lógico a las características propias de la base de datos, se muestra a continuación el modelo físico.



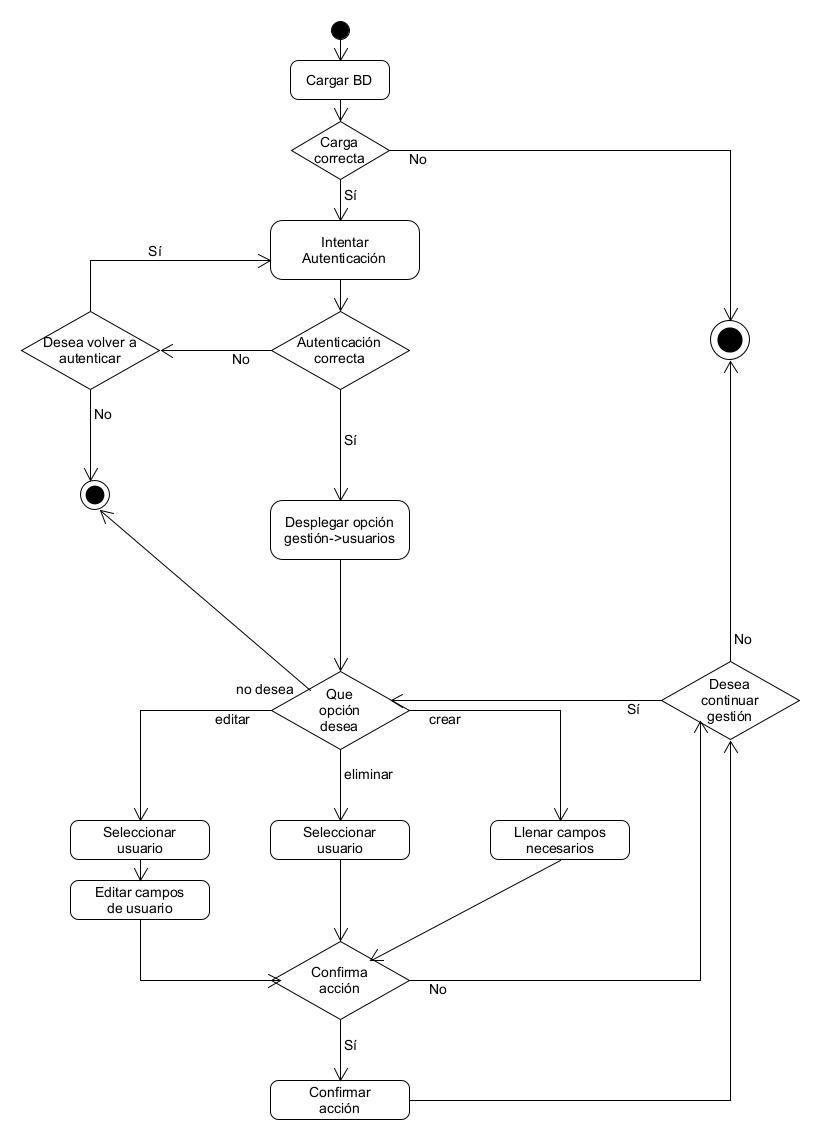
**Figura 5 Modelo físico de los datos.**

A continuación se explican algunas consideraciones hechas al modelo físico de los datos para mejorar la gestión y evitar futuros problemas sobre la gestión de la base de datos.

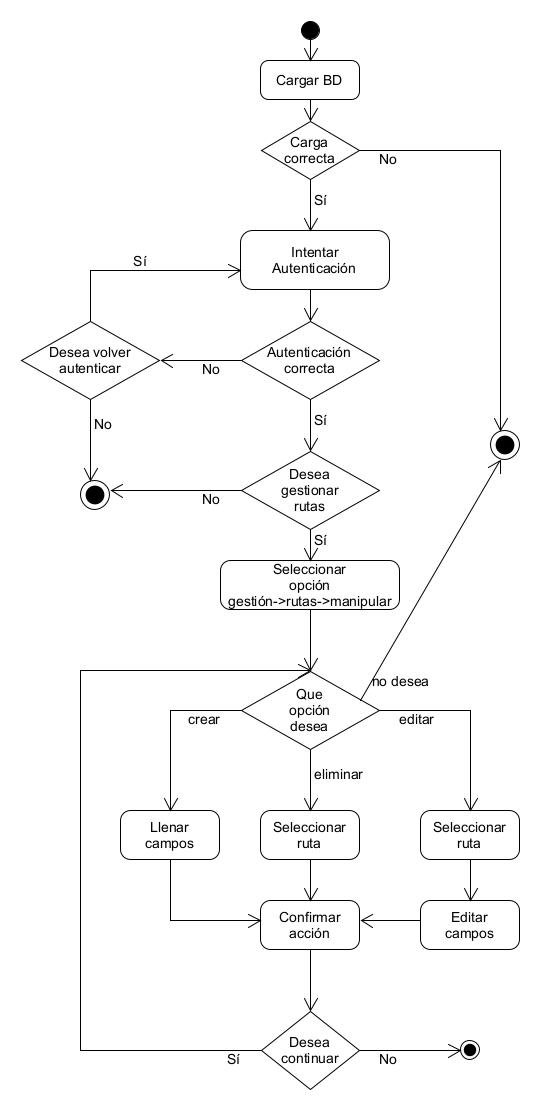
Para evitar problemas de sincronización en entornos distribuidos se estableció como tipo de dato de la llave primaria de Signal el tipo Text, ya que se asigna como identificador por parte del cliente de la base de datos un Identificador Único Universal (UUID, por sus siglas en inglés), ya que este va a ser un elemento frecuente de registrar en la base de datos, y si se utiliza una base de datos SQLite en un entorno distribuido donde múltiples instancias de la base de datos pueden insertar registros en la misma tabla, puede haber problemas de sincronización con las llaves primarias autoincrementales. Esto puede resultar en conflictos y errores al intentar insertar registros con la misma llave primaria. Además de que el uso de AUTOINCREMENT impone un costo adicional en términos de CPU, memoria, espacio en disco y E/S de disco, y debe evitarse si no es estrictamente necesario (Developers, 2021).

## 2.3 Diagramas de actividades.

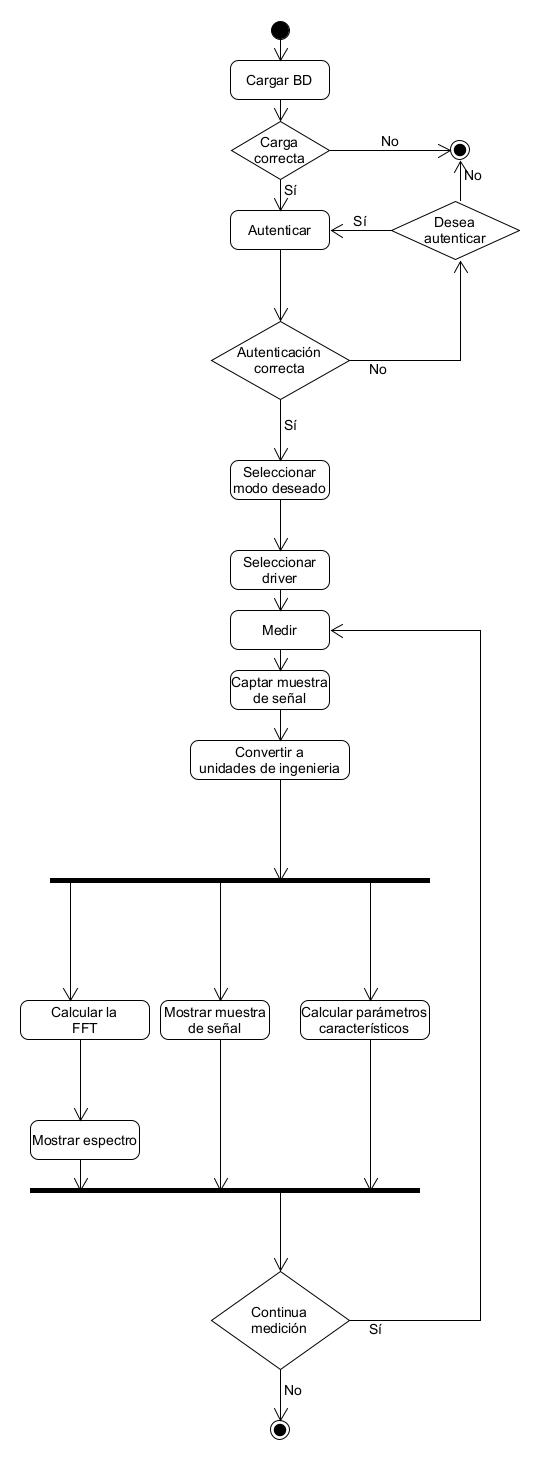
Un diagrama de actividades es una representación gráfica de los flujos de trabajo de actividades y acciones con soporte para elección, iteración y concurrencia. En el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), los diagramas de actividades se utilizan para modelar tanto procesos computacionales como organizacionales (es decir, flujos de trabajo), así como los flujos de datos que se cruzan con las actividades relacionadas (Paradigm, 2021). A continuación se muestran los diagramas de actividades de algunas acciones en la aplicación.



**Figura 6 Diagrama de actividades de la gestión de usuarios.**



**Figura 7 Diagrama de actividades de la gestión de rutas.**



**Figura 8 Diagrama de actividades del proceso de medición.**

## 2.4 Seguridad, roles y privilegios.

Para controlar el acceso a los datos, se debe configurar una estructura organizativa encargada de proteger los datos confidenciales y que, a la vez, permita la colaboración. Para cumplir con este requisito se asignan usuarios a roles de seguridad (Microsoft, 2023).

Un rol de seguridad define cuántos usuarios distintos tienen acceso a diferentes tipos de registros. Para controlar el acceso a los datos, puede modificar los roles de seguridad existentes, crear nuevos o cambiar cuáles se asignan a los usuarios. Un usuario puede tener varios roles de seguridad (Microsoft, 2023).

En el caso específico se establecen estos roles con sus privilegios:

* Administrador: Usuario encargado de la implementación, ajuste y preservación del programa de administración de bases de datos, la creación y administración de cuentas y autorizaciones, el diseño y mejora de la estructura de la base de datos, la ejecución de respaldos y restauración en caso de errores, y la supervisión del desempeño del sistema, gestionar usuarios, configurar parámetros de conexión a la base de datos. En resumen solo puede realizar acciones de gestión de la base de datos, dentro esto se encuentra gestión de usuarios, máquinas, rutas, señales y configuraciones.
* Especialista: Usuario con las habilidades y conocimientos necesarios para realizar el monitoreo, registrar mediciones en la base de datos, configurar parámetros de medición, ya sea, escoger driver y, además de gestionar las rutas y máquinas, realizar y visualizar los análisis de tendencia. En resumen este presenta privilegios de administrador excepto para la gestión de usuarios.
* Visitante: Usuario con la capacidad de ingresar a la aplicación, pero solo puede visualizar las mediciones, tanto la señal en sí, como el espectro. Este rol no puede realizar ninguna acción de gestión de la base de datos, ya sea gestión de señales, usuarios, máquinas o rutas. En resumen el único privilegio que este presenta es la visualización del monitoreo.

Una decisión tomada para mejorar la seguridad de los datos y acceso a la aplicación es la encriptación de las contraseñas mediante el algoritmo de encriptación Secure Hash Algorithm 256 (SHA-256), este es una función hash criptográfica de 256 bits del Secure Hash Algorithm 2 (SHA-2), creada por la Agencia de Seguridad Nacional estadounidense (NSA, por sus siglas en inglés) en 2001.

A continuación se dan a conocer algunas ventajas de SHA-256 que posibilitaron su elección (Gilbert & Handschuh, 2003):

1. Tiene una longitud de salida de 256 bits, lo que le da una alta seguridad contra ataques de búsqueda exhaustiva (fuerza bruta) y de colisiones (encontrar dos mensajes distintos que produzcan el mismo hash).
2. Tiene una estructura interna simple y eficiente, basada en operaciones lógicas y aritméticas sobre palabras de 32 bits. Esto facilita su implementación en hardware y software, y reduce el riesgo de errores o debilidades.
3. Tiene una buena difusión de los bits del mensaje original, lo que implica que un pequeño cambio en el mensaje produce un gran cambio en el hash resultante. Esto dificulta el análisis diferencial y la construcción de mensajes con propiedades especiales.
4. Tiene una buena resistencia a los ataques criptoanalíticos conocidos, como los ataques de cumpleaños, los ataques de preimagen, los ataques de extensión de longitud y los ataques algebraicos. Estos ataques intentan encontrar debilidades en la función hash o en su diseño para reducir el espacio de búsqueda o encontrar colisiones más fácilmente.

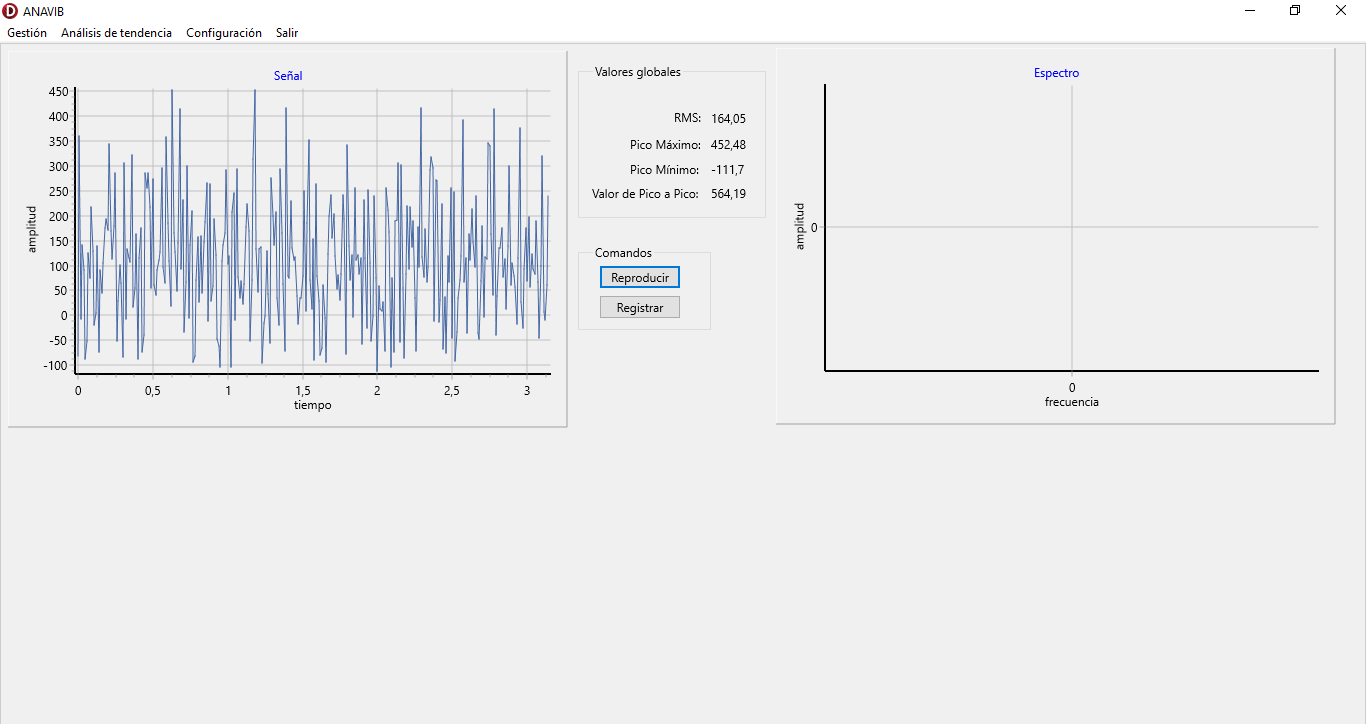
## 2.5 Diseño de la interfaz.

El diseño de interfaz implica la planificación y creación de la apariencia y funcionalidad de la interfaz de usuario de un sistema de software. Esto incluye la definición de cómo el usuario interactúa con la aplicación y cómo se muestra la información. Además, es importante considerar aspectos clave de la experiencia del usuario, como la organización y presentación de la información, el flujo de trabajo, la disposición de las funciones, la legibilidad y la facilidad de navegación (Pressman & Maxim, 2020).

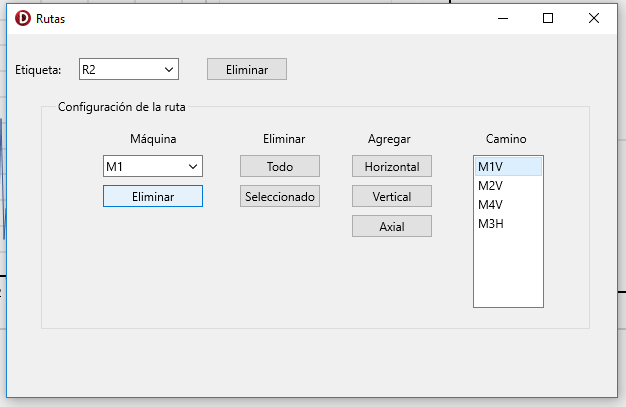
Específicamente, en este proyecto, se respetan los estándares establecidos incluyendo las Reglas de Oro y se cumplen una serie de buenas prácticas mencionadas a continuación (Pressman & Maxim, 2020):

* El diseño es propicio para una fácil interpretación y eficiencia.
* Las operaciones son sencillas de localizar e iniciar.
* Los movimientos son económicos, o sea, requieren muy pocos clics y las distancias son cortas.
* No posee errores ortográficos.
* La interfaz es consistente, sin sobrecarga de la memoria del usuario y brindando el control sobre la aplicación en todo momento.
* Los formularios contienen nombres únicos y sugerentes.
* Permite el flujo principal del trabajo satisfactoriamente.
* Se emplea el mismo tipo de letra y un tamaño acorde, resaltando con negrita y/o color alguna funcionalidad específica.
* El uso del color es moderado, consistente, acorde con el contexto y su significado, sin abuso de imágenes ni gráficos.
* Se previene la ocurrencia de errores, chequeando las entradas, restringiendo los campos, alertando de las posibles consecuencias y permitiendo su corrección mediante mensajes breves y precisos.

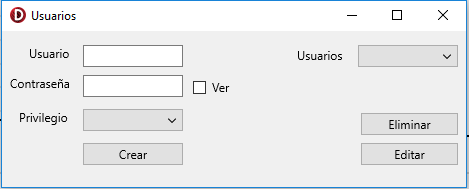
A continuación, se muestran algunas de las interfaces principales del software:



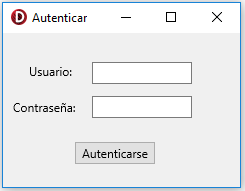
**Figura 9 Ventana principal de la aplicación.**



**Figura 10 Ventana de gestión de las rutas.**



**Figura 11 Ventana para la gestión de usuarios.**



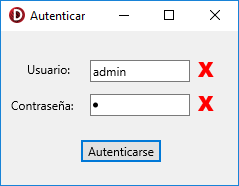
**Figura 12 Ventana para la autenticación de usuarios.**

## 2.6 Tratamiento de errores.

La gestión de errores es un paso esencial para asegurar la precisión y fiabilidad tanto de los datos guardados como del programa en sí. Los errores pueden originarse por una variedad de razones, como errores en la introducción de datos, problemas en el hardware o software, y fallos humanos. Estas inconsistencias pueden causar dificultades tales como registros duplicados, información incompleta o incorrecta, pérdida total de datos, problemas en el funcionamiento del programa, entre otros. Para prevenir estas discrepancias, es necesario llevar a cabo un proceso de detección, corrección y prevención de errores (Lenarduzzi, Saarimäki, & Taibi, 2020).

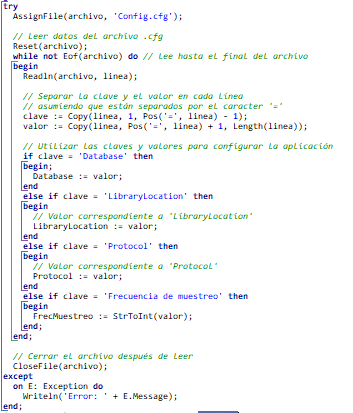
El principal objetivo es evitar los errores, pero una vez estos ocurran tratarlos de la manera más adecuada posible. Los errores pueden evitarse a través de validaciones de la interfaz, y pueden manejarse mediante el uso y captura de excepciones, además del uso de dispadores en SQLite.

En la [Figura 13](#Figura13) se muestra la validación de interfaz y mostrando al usuario en que se equivoca, tratando el error de forma adecuada.



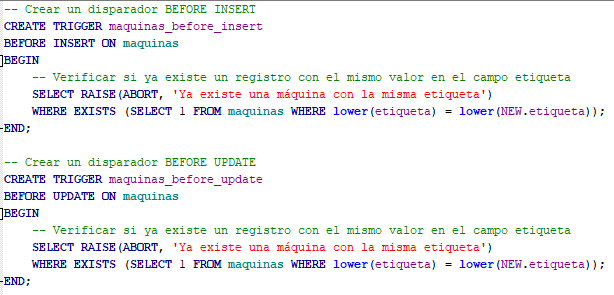
**Figura 13 Tratamiento de errores en la entrada de datos.**

En la [Figura 14](#Figura14) se muestra un ejemplo de evitar errores usando excepciones en Pascal de tratar errores:



**Figura 14 Captura de excepciones.**

En la [Figura 15](#disparador), se visualiza un ejemplo de disparador que evita que se inserte en la tabla máquinas, específicamente en el campo etiqueta, otra máquina con la misma etiqueta, incluyendo el caso que sea escrita en mayúsculas o minúsculas.



**Figura 15 Disparadores BEFORE INSERT y BEFORE UPDATE.**

## Conclusiones parciales.

Con la culminación de este capítulo se puede llegar a las siguientes conclusiones:

1. Se propone una aplicación multiplataforma que permite el monitoreo y que contribuye al diagnóstico de procesos industriales, con la ayuda de un *driver* y el algoritmo de cálculo de la FFT, y el registro en la base de datos se posibilita el análisis de tendencia.
2. El diseño de la base de dato es simple y consistente posibilitando la realización de consultas rápidas, eficientes y eficaces.
3. La seguridad del software se posibilita con la creación de los roles de administrador, especialista y visitante, con sus privilegios y restricciones necesarios. Las contraseñas de usuario nunca son capturadas ni almacenadas en el formato original, sino encriptadas con la función hash SHA-256.
4. El diseño de la interfaz cumple con los estándares, reglas de oro y principios de desarrollo de interfaces, además de que contribuyen al cumplimiento de la visualización correcta del flujo de trabajo.
5. Los errores son importantes de evitar y manejar, estos son evitables mediante la validación de interfaces y manejables mediante excepciones y disparadores.

# Capítulo 3: Validación de la solución

En este capítulo se va a realizar la validación de la solución que consiste en hacer las pruebas de caja negra estudiadas en la asignatura diseño de interfaces y pruebas (DIP).

## 3.1 Pruebas de caja negra.

Las pruebas de caja negra son un método de prueba de software que se basa en los requisitos y especificaciones del sistema, sin tener en cuenta la estructura interna o el código del programa. Este tipo de prueba se centra en verificar si el sistema cumple con las expectativas del usuario y produce los resultados correctos para un conjunto dado de entradas (Nidhra & Yanamadala, 2012).

Las pruebas de caja negra sirven para detectar errores en la funcionalidad, la interfaz de usuario, la integración, la seguridad y el rendimiento del sistema, también pueden ayudar a mejorar la calidad y la confiabilidad del software, así como a reducir el tiempo y el costo de desarrollo y mantenimiento (Nidhra & Yanamadala, 2012).

### 3.1.1 Insertar usuario HU 1.

**Tabla 1 Historia de usuario 1 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Nombre HU** | | **Prioridad** | | **Puntos estimados** |
| HU 1 | Insertar Usuario | |  | |  |
| **Usuario:** Administrador | | | | | |
| **Descripción:** El administrador desea introducir un nuevo usuario para que pueda acceder a la aplicación. | | | | | |
| **Programador responsable:** César Fernández García. | | | | | |
| **Escenarios** | | | | | |
| **Nombre** | | **Condición** | | **Resultado esperado** | |
| Inserción exitosa | | nombre !=null AND contraseña !=null AND NOT EXIST (nombre== usuario.nombre en usuarios) | | Se guarda la información en el listado de usuarios y se muestra el mensaje “Usuario añadido correctamente”. | |
| Datos erróneos | | nombre ==null OR contraseña ==null | | Se emite un mensaje de error: “Datos erróneos”. | |
| Usuario existente en el sistema | | EXIST (nombre == usuario.nombre en usuarios) | | Se advierte al usuario a través de un mensaje que ya existe un usuario con ese nombre en el sistema. | |
| **Variables** | | | | | |
| **Nombre** | | **Tipo o Dominio** | | **Descripción** | |
| nombre | | Cadena | | Nombre de usuario. | |
| contraseña | | Cadena | | Contraseña del usuario. | |
| categoría | | Enumerado={ADMINISTRADOR, ESPECIALISTA, VISITANTE} | | Define los privilegios de usuario. | |
| usuario | | Objeto con atributos (nombre de tipo cadena, contraseña de tipo cadena, privilegio de tipo Enumerado={ADMINISTRADOR, ESPECIALISTA, VISITANTE}) | | Cada usuario de la aplicación. | |
| usuarios | | Conjunto de usuarios. | | Todos los usuarios de la aplicación. | |
| **Iteración asignada** | |  | | | |
| **Observaciones** | |  | | | |

**Casos de prueba de HU 1:**

* Escenario **Inserción exitosa** CP1: ((“admin”, “123”, ADMINISTRADOR), [(“admin2”, “123\*”, ADMINISTRADOR)]).
* Escenario **Datos erróneos** CP2: ((null, “123”, ADMINISTRADOR), [(“admin2”, “123\*”, ADMINISTRADOR)]).
* Escenario **Usuario existente en el sistema** CP3: ((“admin2”, “123”, ADMINISTRADOR), [(“admin2”, “123”, ADMINISTRADOR)]).

**Tabla 2 Casos de prueba de HU 1 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO DE PRUEBA** | | | | | |
| **Historia de Usuario** | | HU 1 | | | |
| **Desarrollador** | | César Fernández García. | | | |
| **Probador** | | Jean Michel Ruiz Remis | | | |
| **Fecha** | | 16/julio/2023 | | | |
| **Combinaciones de valores de entrada** | | | **Técnicas de diseño aplicada** | **Resultados Esperados** | **Resultados**  **Reales** |
| **CP** | **Variable** | **Valor** |
| 1 | nombre | admin | Particiones de equivalencia | Se muestra el mensaje:  “Usuario añadido correctamente”. |  |
| contraseña | 123 |
| categoría | ADMINISTRADOR |
| 2 | nombre | null | Particiones de equivalencia | Mensaje de error:  “Datos erróneos”. |  |
| contraseña | 123 |
| categoría | ADMINISTRADOR |
| 3 | nombre | admin2 | Particiones de equivalencia | Mensaje de advertencia:  “Ya existe ese usuario en el sistema”. |  |
| contraseña | 123 |
| categoría | ADMINISTRADOR |
| **Observaciones:** | | | | | |

### 3.1.2 Editar usuario HU 2.

**Tabla 3 Historia de usuario 2 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Nombre HU** | | **Prioridad** | | **Puntos estimados** |
| HU 2 | Editar Usuario | |  | |  |
| **Usuario:** Administrador | | | | | |
| **Descripción:** El administrador desea editar un usuario que ya existe en la aplicación. | | | | | |
| **Programador responsable:** César Fernández García. | | | | | |
| **Escenarios** | | | | | |
| **Nombre** | | **Condición** | | **Resultado esperado** | |
| Edición exitosa | | nombre !=null AND contraseña !=null AND NOT EXIST (nombre== usuario.nombre en usuarios) | | Se edita la información del usuario y se muestra el mensaje “Usuario editado correctamente”. | |
| Datos erróneos | | nombre ==null OR contraseña ==null | | Se emite un mensaje de error: “Datos erróneos”. | |
| Usuario existente en el sistema | | EXIST (nombre == usuario.nombre en usuarios) | | Se advierte al usuario a través de un mensaje que ya existe un usuario con ese nombre en el sistema. | |
| **Variables** | | | | | |
| **Nombre** | | **Tipo o Dominio** | | **Descripción** | |
| nombre | | Cadena | | Nombre de usuario. | |
| contraseña | | Cadena | | Contraseña del usuario. | |
| categoría | | Enumerado={ADMINISTRADOR, ESPECIALISTA, VISITANTE} | | Define los privilegios de usuario. | |
| usuario | | Objeto con atributos (nombre de tipo cadena, contraseña de tipo cadena, privilegio de tipo Enumerado={administrador, especialista, visitante}) | | Cada usuario de la aplicación. | |
| usuarios | | Conjunto de usuarios. | | Todos los usuarios de la aplicación. | |
| **Iteración asignada** | |  | | | |
| **Observaciones** | |  | | | |

**Casos de prueba de HU 2:**

**Objetivo de edición:** (“admin”, “123”, ADMINISTRADOR)

* Escenario **Edición exitosa** CP1: ((“admin”, “123\*”, ADMINISTRADOR), [(“admin2”, “123\*”, ADMINISTRADOR), (“admin”, “123”, ADMINISTRADOR)]).
* Escenario **Datos erróneos** CP2: ((null, “123\*”, ADMINISTRADOR), [(“admin2”, “123\*”, ADMINISTRADOR), (“admin”, “123”, ADMINISTRADOR)]).
* Escenario **Usuario existente en el sistema** CP3: ((“admin2”, “123\*”, ADMINISTRADOR), [(“admin2”, “123\*”, ADMINISTRADOR), (“admin”, “123”, ADMINISTRADOR)]).

**Tabla 4 Casos de prueba de HU 2 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO DE PRUEBA** | | | | | |
| **Historia de Usuario** | | HU 2 | | | |
| **Desarrollador** | | César Fernández García. | | | |
| **Probador** | | Jean Michel Ruiz Remis | | | |
| **Fecha** | | 16/julio/2023 | | | |
| **Combinaciones de valores de entrada** | | | **Técnicas de diseño aplicada** | **Resultados Esperados** | **Resultados**  **Reales** |
| **CP** | **Variable** | **Valor** |
| 1 | nombre | admin | Particiones de equivalencia | Se muestra el mensaje:  “Usuario editado correctamente”. |  |
| contraseña | 123\* |
| categoría | ADMINISTRADOR |
| 2 | nombre | null | Particiones de equivalencia | Mensaje de error:  “Datos erróneos”. |  |
| contraseña | 123 |
| categoría | ADMINISTRADOR |
| 3 | nombre | admin2 | Particiones de equivalencia | Mensaje de advertencia:  “Ya existe ese usuario en el sistema”. |  |
| contraseña | 123 |
| categoría | ADMINISTRADOR |
| **Observaciones:** | | | | | |

### 3.1.3 Eliminar usuario HU 3.

**Tabla 5 Historia de usuario 3 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Nombre HU** | | **Prioridad** | | **Puntos estimados** |
| HU 3 | Eliminar usuario | |  | |  |
| **Usuario:** Administrador | | | | | |
| **Descripción:** El administrador desea eliminar un usuario que ya existe en la aplicación. | | | | | |
| **Programador responsable:** César Fernández García. | | | | | |
| **Escenarios** | | | | | |
| **Nombre** | | **Condición** | | **Resultado esperado** | |
| Eliminación exitosa | | nombre !=null AND nombre != usuario\_autenticado AND categoría == ADMINISTRADOR AND( LENGTH(administradores) > 1) | | Se elimina al usuario del sistema “Usuario eliminado correctamente”. | |
| Datos erróneos | | nombre ==null | | Se emite un mensaje de error: “Datos erróneos”. | |
| Usuario inexistente en el sistema | | NOT EXIST (nombre == usuario.nombre en usuarios) | | Se advierte al usuario a través de un mensaje que “No existe un usuario con ese nombre en el sistema”. | |
| **Variables** | | | | | |
| **Nombre** | | **Tipo o Dominio** | | **Descripción** | |
| nombre | | Cadena | | Nombre de usuario. | |
| usuario | | Objeto con atributos (nombre de tipo cadena, contraseña de tipo cadena, privilegio de tipo Enumerado={administrador, especialista, visitante}) | | Cada usuario de la aplicación. | |
| usuarios | | Conjunto de usuarios. | | Todos los usuarios de la aplicación. | |
| **Iteración asignada** | |  | | | |
| **Observaciones** | |  | | | |

**Casos de prueba de HU 3:**

**Objetivo de eliminación:** (“admin”, “123”, ADMINISTRADOR)

* Escenario **Eliminación exitosa** CP1: ((“admin”), [(“admin2”, “123\*”, ADMINISTRADOR), (“admin”, “123”, ADMINISTRADOR)]).
* Escenario **Datos erróneos** CP2: ((null, “123”, ADMINISTRADOR), [(“admin2”, “123\*”, ADMINISTRADOR), (“admin”, “123”, ADMINISTRADOR)]).
* Escenario **Usuario inexistente en el sistema** CP3: ((“admin3”, “123”, ADMINISTRADOR), [(“admin2”, “123\*”, ADMINISTRADOR), (“admin”, “123”, ADMINISTRADOR)]).

**Tabla 6 Casos de prueba de HU 3 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO DE PRUEBA** | | | | | |
| **Historia de Usuario** | | HU 3 | | | |
| **Desarrollador** | | César Fernández García. | | | |
| **Probador** | | Jean Michel Ruiz Remis | | | |
| **Fecha** | | 16/julio/2023 | | | |
| **Combinaciones de valores de entrada** | | | **Técnicas de diseño aplicada** | **Resultados Esperados** | **Resultados**  **Reales** |
| **CP** | **Variable** | **Valor** |
| 1 | nombre | admin | Particiones de equivalencia | Se muestra el mensaje:  “Usuario eliminado correctamente”. |  |
| 2 | nombre | null | Particiones de equivalencia | Mensaje de error:  “Datos erróneos”. |  |
| 3 | nombre | admin3 | Particiones de equivalencia | Mensaje de advertencia:  “No existe ese usuario en el sistema”. |  |
| **Observaciones:** | | | | | |

### 3.1.4 Insertar ruta HU 4.

**Tabla 7 Historia de usuario 4 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Nombre HU** | | **Prioridad** | | **Puntos estimados** |
| HU 4 | Insertar ruta | |  | |  |
| **Usuario:** Especialista | | | | | |
| **Descripción:** El especialista desea introducir una nueva ruta para usar en la aplicación. | | | | | |
| **Programador responsable:** César Fernández García. | | | | | |
| **Escenarios** | | | | | |
| **Nombre** | | **Condición** | | **Resultado esperado** | |
| Inserción exitosa | | etiqueta !=null AND NOT EXIST (etiqueta== ruta.etiqueta en rutas) | | Se guarda la información en el listado de rutas y se muestra el mensaje “Ruta añadida correctamente”. | |
| Datos erróneos | | etiqueta ==null | | Se emite un mensaje de error: “Datos erróneos”. | |
| Ruta existente en el sistema | | EXIST (etiqueta == ruta.etiqueta en rutas) | | Se advierte al usuario a través de un mensaje que ya existe una ruta con esa etiqueta en el sistema. | |
| **Variables** | | | | | |
| **Nombre** | | **Tipo o Dominio** | | **Descripción** | |
| etiqueta | | Cadena | | Nombre de ruta. | |
| camino | | Cadena | | Trayectoria de la ruta. | |
| ruta | | Objeto con atributos (etiqueta de tipo cadena, camino de tipo cadena) | | Cada ruta de la aplicación. | |
| rutas | | Conjunto de rutas. | | Todas las rutas de la aplicación. | |
| **Iteración asignada** | |  | | | |
| **Observaciones** | |  | | | |

**Casos de prueba de HU 4:**

* Escenario **Inserción exitosa** CP1: ((“R1”, “M1H, M2V”), [(“R2”, “M1H, M2V”)]).
* Escenario **Datos erróneos** CP2: ((null, “M1H, M2V”), [(“R2”, “M1H, M2V”)]).
* Escenario **Ruta existente en el sistema** CP3: ((“R2”, “M1H, M2V”)), [(“R2”, “M1H, M2V”)]).

**Tabla 8 Casos de prueba de HU 4 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO DE PRUEBA** | | | | | |
| **Historia de Usuario** | | HU 4 | | | |
| **Desarrollador** | | César Fernández García. | | | |
| **Probador** | | Jean Michel Ruiz Remis | | | |
| **Fecha** | | 16/julio/2023 | | | |
| **Combinaciones de valores de entrada** | | | **Técnicas de diseño aplicada** | **Resultados Esperados** | **Resultados**  **Reales** |
| **CP** | **Variable** | **Valor** |
| 1 | etiqueta | admin | Particiones de equivalencia | Se muestra el mensaje:  “Usuario añadido correctamente”. |  |
| camino | 123 |
| 2 | etiqueta | null | Particiones de equivalencia | Mensaje de error:  “Datos erróneos”. |  |
| camino | 123 |
| 3 | etiqueta | admin2 | Particiones de equivalencia | Mensaje de advertencia:  “Ya existe ese usuario en el sistema”. |  |
| camino | 123 |
| **Observaciones:** | | | | | |

### 3.1.5 Insertar máquina HU 5.

**Tabla 9 Historia de usuario 5 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Nombre HU** | | **Prioridad** | | **Puntos estimados** |
| HU 5 | Insertar máquina | |  | |  |
| **Usuario:** Especialista | | | | | |
| **Descripción:** El especialista desea introducir una nueva máquina para usar en la aplicación. | | | | | |
| **Programador responsable:** César Fernández García. | | | | | |
| **Escenarios** | | | | | |
| **Nombre** | | **Condición** | | **Resultado esperado** | |
| Inserción exitosa | | etiqueta !=null AND NOT EXIST (etiqueta== máquina.etiqueta en máquinas) | | Se guarda la información en el listado de máquinas y se muestra el mensaje “Máquina añadida correctamente”. | |
| Datos erróneos | | etiqueta ==null | | Se emite un mensaje de error: “Datos erróneos”. | |
| Máquina existente en el sistema | | EXIST (etiqueta == máquina.etiqueta en máquinas) | | Se advierte al usuario a través de un mensaje que ya existe una máquina con esa etiqueta en el sistema. | |
| **Variables** | | | | | |
| **Nombre** | | **Tipo o Dominio** | | **Descripción** | |
| etiqueta | | Cadena | | Nombre de la máquina. | |
| máquina | | Objeto con atributos (etiqueta de tipo cadena) | | Cada máquina de la aplicación. | |
| máquinas | | Conjunto de máquinas. | | Todas las máquinas de la aplicación. | |
| **Iteración asignada** | |  | | | |
| **Observaciones** | |  | | | |

**Casos de prueba de HU 5:**

* Escenario **Inserción** **exitosa** CP1: ((“M1”), [(“M2”), (“M3”)]).
* Escenario **Datos erróneos** CP2: ((null), [(“M2”), (“M3”)]).
* Escenario **Máquina existente en el sistema** CP3: ((“M2”), [(“M2”), (“M3”)]).

**Tabla 10 Casos de prueba de HU 5 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO DE PRUEBA** | | | | | |
| **Historia de Usuario** | | HU 5 | | | |
| **Desarrollador** | | César Fernández García. | | | |
| **Probador** | | Jean Michel Ruiz Remis | | | |
| **Fecha** | | 16/julio/2023 | | | |
| **Combinaciones de valores de entrada** | | | **Técnicas de diseño aplicada** | **Resultados Esperados** | **Resultados**  **Reales** |
| **CP** | **Variable** | **Valor** |
| 1 | etiqueta | M1 | Particiones de equivalencia | Se muestra el mensaje:  “Máquina ingresada correctamente”. |  |
| 2 | etiqueta | null | Particiones de equivalencia | Mensaje de error:  “Datos erróneos”. |  |
| 3 | etiqueta | M2 | Particiones de equivalencia | Mensaje de advertencia:  “Ya existe esa máquina en el sistema”. |  |
| **Observaciones:** | | | | | |

### 3.1.6 Insertar señal HU 6.

**Tabla 11 Historia de usuario 6 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Nombre HU** | | **Prioridad** | | **Puntos estimados** |
| HU 6 | Insertar señal | |  | |  |
| **Usuario:** Especialista | | | | | |
| **Descripción:** El especialista desea introducir una nueva señal para usar en la aplicación. | | | | | |
| **Programador responsable:** César Fernández García. | | | | | |
| **Escenarios** | | | | | |
| **Nombre** | | **Condición** | | **Resultado esperado** | |
| Inserción exitosa | | señal\_datos !=null | | Se guarda la información en el listado de señales y se muestra el mensaje “Señal añadida correctamente”. | |
| Datos erróneos | | señal\_datos ==null | | Se emite un mensaje de error: “Error de medición”. | |
| **Variables** | | | | | |
| **Nombre** | | **Tipo o Dominio** | | **Descripción** | |
| señal\_datos | | Arreglo de float | | Cada coordenada en el eje cartesiano. | |
| señal | | Objeto con atributos (señal\_datos de tipo arreglo de float) | | Cada señal de la aplicación. | |
| señales | | Conjunto de rutas. | | Todas las señales de la aplicación. | |
| **Iteración asignada** | |  | | | |
| **Observaciones** | |  | | | |

**Casos de prueba de HU 6:**

* Escenario **Inserción** **exitosa** CP1: (([3.12, 5.12, 45.2]), [([3.14, 5.17, 45.4]), ([3.5, 5.6, 8.9])]).
* Escenario **Datos erróneos** CP2: ((null), [([3.14, 5.17, 45.4]), ([3.5, 5.6, 8.9])]).

**Tabla 12 Casos de prueba de HU 6 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO DE PRUEBA** | | | | | |
| **Historia de Usuario** | | HU 6 | | | |
| **Desarrollador** | | César Fernández García. | | | |
| **Probador** | | Jean Michel Ruiz Remis | | | |
| **Fecha** | | 16/julio/2023 | | | |
| **Combinaciones de valores de entrada** | | | **Técnicas de diseño aplicada** | **Resultados Esperados** | **Resultados**  **Reales** |
| **CP** | **Variable** | **Valor** |
| 1 | señal\_datos | [3.12,5.12,45.2] | Particiones de equivalencia | Se muestra el mensaje:  “Señal registrada correctamente”. |  |
| 2 | señal\_datos | null | Particiones de equivalencia | Mensaje de error:  “Datos erróneos”. |  |
| **Observaciones:** | | | | | |

### 3.1.7 Autenticar HU 7.

**Tabla 13 Historia de usuario 7 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Identificador** | **Nombre HU** | | **Prioridad** | | **Puntos estimados** |
| HU 7 | Autenticar | |  | |  |
| **Usuario:** Administrador, Especialista y Visitante. | | | | | |
| **Descripción:** Cualquier usuario que desea autenticarse en la aplicación. | | | | | |
| **Programador responsable:** César Fernández García. | | | | | |
| **Escenarios** | | | | | |
| **Nombre** | | **Condición** | | **Resultado esperado** | |
| Autenticación exitosa | | nombre !=null AND contraseña !=null AND EXIST (nombre == usuario.nombre AND contraseña == usuario. contraseña) | | Se guarda la información en el listado de usuarios y se muestra el mensaje “Usuario autenticado correctamente”. | |
| Datos erróneos | | nombre !=null OR contraseña !=null OR NOT EXIST (nombre == usuario.nombre AND contraseña == usuario. contraseña) | | Se emite un mensaje de error: “Error de autenticación”. | |
| **Variables** | | | | | |
| **Nombre** | | **Tipo o Dominio** | | **Descripción** | |
| nombre | | Cadena | | Nombre del usuario. | |
| contraseña | | Cadena | | Contraseña de autenticación del usuario. | |
| usuario | | Objeto con atributos (nombre de tipo cadena) | | Cada usuario de la aplicación. | |
| usuarios | | Conjunto de usuarios. | | Todos los usuarios de la aplicación. | |
| **Iteración asignada** | |  | | | |
| **Observaciones** | |  | | | |

**Casos de prueba de HU 7:**

* Escenario **Autenticación exitosa** CP1: (([3.12, 5.12, 45.2]), [([3.14, 5.17, 45.4]), ([3.5, 5.6, 8.9])]).
* Escenario **Datos erróneos** CP2: ((null), [([3.14, 5.17, 45.4]), ([3.5, 5.6, 8.9])]).

**Tabla 14 Casos de prueba de HU 7 (Caja negra).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO DE PRUEBA** | | | | | |
| **Historia de Usuario** | | HU 7 | | | |
| **Desarrollador** | | César Fernández García. | | | |
| **Probador** | | Jean Michel Ruiz Remis | | | |
| **Fecha** | | 16/julio/2023 | | | |
| **Combinaciones de valores de entrada** | | | **Técnicas de diseño aplicada** | **Resultados Esperados** | **Resultados**  **Reales** |
| **CP** | **Variable** | **Valor** |
| 1 | nombre | “admin” | Particiones de equivalencia | Se muestra el mensaje:  “Usuario autenticado correctamente”. |  |
| contraseña | “123” |
| 2 | nombre | null | Particiones de equivalencia | Mensaje de error:  “Datos erróneos”. |  |
| contraseña | “123” |
| **Observaciones:** | | | | | |

## Conclusiones parciales.

1. Las pruebas de caja negra permiten detectar errores de interfaz, funcionalidad y de seguridad.
2. Con las validaciones realizadas y comprobadas se valida que la solución propuesta satisface los principios de funcionamiento de la aplicación.

# Conclusiones

Luego del proceso investigativo y de desarrollo se puede concluir que:

1. Los sistemas de monitoreo y diagnóstico permiten la determinación del estado de procesos y máquinas industriales.
2. La utilización de la función hash SHA-256 y de la autenticación de usuarios, permiten la trasmisión de contraseñas de forma cifrada y evita el uso del software por usuarios indebidos, lo cual garantiza seguridad y confidencialidad.
3. La información visual de las señales y espectro brinda apoyo al proceso de toma de decisiones en el diagnóstico de máquinas industriales.
4. La utilización de principios de diseño garantizan la reutilización y mantenibilidad de la herramienta.

# Recomendaciones

Posterior al proceso de diseño, desarrollo e implementación de este sistema, se han recogido un conjunto de sugerencias que permitirán un mejor desempeño de la aplicación en cuestión.

1. Realizar periódicamente pruebas regulares para la detección de posibles fallas y realización de mejoras en los protocolos de seguridad, así como proporcionar soporte a la aplicación para asegurar que las funcionalidades sean efectivas.
2. Agregar la funcionalidad de alertas para notificar mensajes emergentes, correo electrónico o un sonido cuando se cumplan sus condiciones.
3. Permitir registrar y visualizar las acciones realizadas en la aplicación.
4. Implementar el uso de la inteligencia artificial para un posible acercamiento al diagnóstico real del equipo industrial.

# Referencias bibliográficas

¿Qué es la Industria 4.0 y cómo funciona? | IBM. (July de 2023). *¿Qué es la Industria 4.0 y cómo funciona? | IBM*. Obtenido de https://www.ibm.com/es-es/topics/industry-4-0

Brüel & Kjær. (July de 2023). Software de análisis de señales. *Software de análisis de señales*. Obtenido de https://www.bksv.com/es/analysis-software/data-acquisition-software/bk-connect/signal-analysis

Cantú, M. (July de 2003). *Essential Pascal.* (M. C. Books, Ed.) Marco Cantú Books.

Davuluru, V. S., Hettiarachchi, D. L., & Balster, E. (2022). Performance Analysis of DFT and FFT Algorithms on Modern GPUs.

de Murga Aguiar, G. A. (September de 2020). *Herramienta de monitoreo y control en tiempo real de los nodos HEATS-RT.* candthesis, Universidad Tecnológica de La Habana José Antonio Echeverría, La.

Delphi 11.3. (July de 2023). *Delphi 11.3*. Obtenido de https://lenguajesdeprogramacion.net/pascal-y-delphi/

Delphi Características. (July de 2023). *Delphi Características*. Obtenido de https://www.embarcadero.com/es/products/delphi/features/delphi

Developers, S. (2021). SQLite Autoincrement. *SQLite Autoincrement*. Obtenido de https://www.sqlite.org/autoinc.html

Dynamox. (July de 2023). Aumente la disponibilidad con Dynamox. *Aumente la disponibilidad con Dynamox*. Obtenido de https://dynamox.net/es

Ferrer, J. (July de 2023). Definición de Monitoreo. *Definición de Monitoreo*. Obtenido de https://enciclopedia.net/monitoreo/

Gilbert, H., & Handschuh, H. (2003). Security analysis of SHA-256 and sisters. *International workshop on selected areas in cryptography*, (págs. 175–193).

Lenarduzzi, V., Saarimäki, N., & Taibi, D. (2020). A systematic literature review on error handling in software development. *Information and Software Technology, 129*, 106377. doi:10.1016/j.infsof.2020.106377

Lenguaje de programación Pascal actualizado a 2023. (July de 2023). *Lenguaje de programación Pascal actualizado a 2023*. Obtenido de https://lenguajesdeprogramacion.net/pascal-y-delphi/

Marín, E. P. (1997). *La medición y el análisis de vibraciones en el diagnóstico de máquinas rotatorias.* (D. de Ingeniería de las vibraciones y diagnóstico, Ed.) Centro de Estudios Innovación y Mantenimiento.

Matos, R. M. (2010). *Sistemas de Base de Datos.* (F. Varela, Ed.) La Habana.

Microsoft. (2023). Controlar el acceso a los datos y recursos mediante roles de seguridad. *Controlar el acceso a los datos y recursos mediante roles de seguridad*. Obtenido de https://docs.microsoft.com/es-es/powerapps/maker/data-platform/data-platform-security-roles

Ministerio de Relaciones Exteriores de Cuba. (July de 2023). Informe de Cuba en virtud de la resolución 75/289 de la Asamblea General de las Naciones Unidas, titulada “Necesidad de poner fin al bloqueo económico, comercial y financiero impuesto por los Estados Unidos de América contra Cuba”. *Informe de Cuba en virtud de la resolución 75/289 de la Asamblea General de las Naciones Unidas, titulada “Necesidad de poner fin al bloqueo económico, comercial y financiero impuesto por los Estados Unidos de América contra Cuba”*. Calle Calzada, No. 360, Vedado. Plaza de la Revolución. La Habana, Cuba. Obtenido de https://cubaminrex.cu/es/informe-de-cuba-en-virtud-de-la-resolucion-75289-de-la-asamblea-general-de-las-naciones-unidas

Mohd Ghazali, M. H., & Rahiman, W. (2021). Vibration analysis for machine monitoring and diagnosis: a systematic review. *Shock and Vibration, 2021*, 1–25.

Nidhra, S., & Yanamadala, M. (2012). Black box and white box testing techniques–a literature review. *International Journal of Embedded Systems and Applications, 2*, 29–50.

PAESSLER. (July de 2023). Supervisión de procesos industriales. *Supervisión de procesos industriales*. Obtenido de https://www.paessler.com/es/industrial-it-monitoring

Paradigm, V. (2021). What is Activity Diagram? *What is Activity Diagram?* Obtenido de https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/

Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner’s Approach* (9 ed.). McGraw-Hill Education.

Ramos Cosio, F. J. (2020). Revisión de los beneficios con la aplicación del mantenimiento predictivo para la mejora en la confiabilidad de las máquinas industriales en el sector industrial en los últimos diez años.

SQLite. (July de 2023). About SQLite. *About SQLite*. Obtenido de https://www.sqlite.org/about.html

SQLite. (July de 2023). Appropriate Uses For SQLite. *Appropriate Uses For SQLite*. Obtenido de https://www.sqlite.org/whentouse.html

SQLite. (July de 2023). Single-file Cross-platform Database. *Single-file Cross-platform Database*. Obtenido de https://www.sqlite.org/onefile.html

SQLite. (July de 2023). System Requirements For SQLite. *System Requirements For SQLite*. Obtenido de https://www.sqlite.org/draft/sysreq.html

SQLite. (July de 2023). What Is SQLite? *What Is SQLite?* Obtenido de https://www.sqlite.org/index.html

Team, S. D. (s.f.). SQLite Is The Most Widely Deployed SQL Database Engine In The World. *SQLite Is The Most Widely Deployed SQL Database Engine In The World.* Obtenido de https://www.sqlite.org/mostdeployed.html

Wikipedia. (July de 2023). Pascal (lenguaje de programación) — Wikipedia, La enciclopedia libre. *Pascal (lenguaje de programación) — Wikipedia, La enciclopedia libre*. Obtenido de https://es.wikipedia.org/w/index.php?title=Pascal\_(lenguaje\_de\_programaci%C3%B3n)&oldid=151221731

Zambrano-Reyes, A., Nossov, V. R., & Gómez-Mancilla, J. C. (s.f.). Análisis de la Respuesta Vibratoria de Ejes Fisurados sobre Chumaceras Lubricadas para Control y Atenuación de Vibraciones en Máquinas Rotatorias.