Facultad 1

Aplicación informática para la adquisición de datos en vehículos mediante el dispositivo OBD2.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Yandry Roger García Pérez

Tutores: Ing. Julio Alberto Leyva Durán

Ing. Vladimir Campos Kindelán

Co-tutor: Ing. Osberto Prieto Pérez.

La Habana, 28 noviembre de 2022

Año 62 de la Revolución

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título "***Aplicación informática para la adquisición de datos en vehículos mediante el dispositivo OBD2***" concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autores de su contenido. Para que así conste firma la presente a los 28 días del mes de noviembre del año 2022.

Yandry Roger García Pérez **Ing. Vladimir Campos Kindelán.**

_____ _____

Firma del Autor Firma del Tutor

Ing. Julio Alberto Leyva Durán.

_____

Firma del Tutor

DATOS DE CONTACTO

Autor:

Yandry Roger García Pérez

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: yandryrgp@estudiantes.uci.cu

Tutores:

Ing. Julio Alberto Leyva Durán.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jaleyva@uci.cu

Ing. Vladimir Campos Kindelán

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: vladimirc@uci.cu

Cotutor:

Ing. Osberto Prieto Pérez.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: oprietop@uci.cu

AGRADECIMIENTOS

"De chico me enseñaron a dar gracias por las cosas buenas (y también malas) de la vida". Por eso, en esta tesis voy a agradecer a:

Mis padres y hermana, que sin ellos nada de esto hubiera sido posible.

Mis profesores.

Mis tutores y los demás trabajadores del proyecto por el apoyo dado.

Gracias a todos

DEDICATORIA

Quiero dedicar el resultado de este trabajo a todos los que influyeron en la realización de este y en mi camino para alcanzar el título de ingeniero en ciencias informáticas.

A mi mamá, que ha dado todo su empeño porque yo logre mis sueños, y que siempre ha estado hay para mí, sin importar mis decisiones.

A mi padre, que se ha sacrificado y me ha enseñado a luchar para conseguir mis metas, ha luchado por mis sueños y ha sido mi modelo de hombre a seguir.

Mi hermana, que ha estado ahí a mi lado, apoyándome siempre.

A mis amigos, los que están aún en mi vida y a los que no, pero que han marcado mi vida y me han ayudado a ser quien soy. En especial a Álvaro, Adiel, Hugo, Javier, Felo, Lenna, Heydi, Neiser quienes fueron mi familia dentro de la universidad.

Daylin, quien desde el primer día fue más que una amiga, una hermana que me ayudó en los momentos malos y buenos, quien me hizo luchar y no rendirme nunca.

Leidy Laura, una de las responsables de mi crecimiento como persona, me enseñó a creer en mí, a no rendirme y seguir para adelante con mis sueños. Gracias a todos y esto es especialmente para ustedes.

RESUMEN

Una de las innovaciones que ha revolucionado la industria automotriz es la inclusión de los sistemas OBD2 (*On Board Diagnostic - Second Generation*), su objetivo principal es el de detectar fallas que se encuentren en el sistema de inyección lo más efectivamente para contrarrestar las emisiones provocadas en la combustión. La presente investigación tiene como objetivo básico la implementación de una aplicación informática para la adquisición de datos en vehículos mediante el dispositivo OBD2. Para el marco de este trabajo se realizó un estudio sobre la evaluación de los procesos para el diagnóstico de vehículos, se modelo una propuesta de solución mediante los artefactos ingenieriles, que propone la metodología AUP-UCI en su escenario 4 y se implementó y valido el sistema, en correspondencia con los requisitos de software extraídos mediante el estudio de las aplicaciones homólogas, mediante entrevistas y encuestas realizadas. Como resultado se obtuvo una aplicación de desktop la cual permite la adquisición de los datos de los vehículos, mediante un prototipo de dispositivo OBD2 utilizando placa Arduino UNO, el cual permite una mejor toma de decisiones en cuanto al mantenimiento de los vehículos y una reducción de la emisión de CO_2 influyendo positivamente el cuidado del medio ambiente.

PALABRAS CLAVE

Adquisición, Arduino, datos, OBD2, vehículo

ABSTRACT

One of the innovations that has revolutionized the automotive industry is the inclusion of OBD2 systems, its main objective is to detect faults found in the injection system as effectively as possible to counteract the emissions caused in combustion. The basic objective of this research is the implementation of a computer application for the acquisition of data in vehicles through the OBD2 device. For the framework of this work, a study was carried out on the evaluation of the processes for the diagnosis of vehicles, a solution proposal was modeled through engineering artifacts, which proposes the AUP-UCI methodology in its scenario 4 and the method was implemented and validated. System, in correspondence with the software requirements extracted through the study of homologous applications, through interviews and surveys carried out. As a result, a desktop application was obtained which allows the acquisition of vehicle data, through a prototype OBD2 device using the Arduino UNO board, which allows better decision-making regarding vehicle maintenance and a reduction in costs. The emission of CO_2 positively influences the care of the environment.

KEYWORDS

Acquisition, Arduino, data, OBD2, vehicle

Índice

INTRODUCCIÓN **11**

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA SOBRE LOS DISPOSITIVOS OBD2 Y LOS SISTEMAS DE ADQUISICIÓN DE DATOS. **15**

1.1 HISTORIA DEL OBD 15

1.2 SISTEMA DE ADQUISICIÓN DE DATOS 20

1.3 APLICACIONES ESCRITORIO 21

1.4 ESTUDIO DE HOMÓLOGOS 22

1.5 PLATAFORMA ARDUINO 24

1.6 AMBIENTE DE DESARROLLO 25

1.7 METODOLOGÍA DE DESARROLLO 28

CONCLUSIONES DEL CAPÍTULO 30

CAPÍTULO 2: PLANIFICACIÓN Y DISEÑO DEL SISTEMA INFORMÁTICO PARA LA ADQUISICIÓN DE DATOS EN VEHÍCULOS MEDIANTE EL DISPOSITIVO OBD2. ..... **31**

2.1 MAPA CONCEPTUAL 31

2.2 LEVANTAMIENTO DE REQUISITOS 32

2.3 ESPECIFICACIÓN DE REQUISITOS 34

2.4 DESCRIPCIÓN DE REQUISITOS DE SOFTWARE 41

2.5 DISEÑO DE LA PROPUESTA DE SOLUCIÓN 42

2.6 DISEÑO DE BASE DATOS 47

CONCLUSIONES DEL CAPÍTULO 49

CAPÍTULO 3: PRUEBAS Y VALIDACIÓN DEL SISTEMA INFORMÁTICO PARA LA ADQUISICIÓN DE DATOS EN VEHÍCULOS MEDIANTE EL DISPOSITIVO OBD2. ..... **50**

3.1 IMPLEMENTACIÓN 50

3.2 MODELO DE COMPONENTES 51

3.3 DIAGRAMA DE DESPLIEGUE 51

3.4 INTERFAZ GRÁFICA DE USUARIO 52

3.5 PRUEBAS DE SOFTWARE 53

CONCLUSIONES DEL CAPÍTULO 63

CONCLUSIONES FINALES **64**

RECOMENDACIONES **65**

REFERENCIAS BIBLIOGRÁFICAS **66**

ANEXOS **75**

ÍNDICE DE TABLAS

TABLA 1 ESTUDIO DE APLICACIONES HOMOLOGAS (FUENTE: ELABORACIÓN PROPIA) 23

TABLA 2 NIVEL DE IMPORTANCIA (FUENTE: ELABORACIÓN PROPIA)..... 35

TABLA 3 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 1 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 36

TABLA 4 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 2 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 37

TABLA 5 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 3 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 37

TABLA 6 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 4 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 38

TABLA 7 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 5 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 39

TABLA 8 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 6 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 39

TABLA 9 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 7 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 40

TABLA 10 SUB CARACTERÍSTICAS DE LA CARACTERÍSTICA 8 DEL MODELO DE CALIDAD EXTERNA, ESPECIFICANDO IMPORTANCIA Y CRITERIO DE EVALUACIÓN (FUENTE: ELABORACIÓN PROPIA) 40

TABLA 11 HU GESTIONAR CONEXIÓN (FUENTE: ELABORACIÓN PROPIA). 41

TABLA 12 DESCRIPCIÓN DE LA BASE DATOS - TABLA INSPECCION..... 47

TABLA 13 DESCRIPCIÓN DE LA BASE DATOS- TABLA FALLOS 48

TABLA 14 DESCRIPCIÓN DE LA BASE DATOS- TABLA VEHÍCULO..... 48

TABLA 15 PRUEBAS DE ACEPTACIÓN HISTORIA DE USUARIO 1 (FUENTE: ELABORACIÓN PROPIA) 55

TABLA 16 PRUEBA DE REGRESIÓN (FUENTE: ELABORACIÓN PROPIA) 56

TABLA 17 PRUEBAS UNITARIAS (FUENTE: ELABORACIÓN PROPIA) 58

TABLA 18 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO, PRUEBAS DE CARGA 60

TABLA 19 *RESULTADOS* DE LAS PRUEBAS DE RENDIMIENTO, PRUEBA DE RESISTENCIA (FUENTE: ELABORACIÓN PROPIA) 60

TABLA 20 RESULTADOS DE LAS PRUEBAS DE RENDIMIENTO, PRUEBA DE RESISTENCIA (FUENTE: ELABORACIÓN PROPIA) 61

TABLA 21 HU GESTIONAR ADQUISICIÓN DE DATOS (FUENTE: ELABORACIÓN PROPIA)..... 76

TABLA 22 HU GESTIONAR DATOS ADQUIRIDOS (FUENTE: ELABORACIÓN PROPIA). 77

TABLA 23 HU VISUALIZAR DATOS EN TIEMPO REAL DEL VEHÍCULO (FUENTE: ELABORACIÓN PROPIA)..... 79

TABLA 24 PRUEBA DE ACEPTACIÓN HISTORIA DE USUARIO 2 (FUENTE: ELABORACIÓN PROPIA) 82

ÍNDICE DE FIGURAS

FIGURA 1 INTERFAZ OBD1 [TOMADA DE: (CÓRDOVA, MAGALLANES 2020)] 16

FIGURA 2 *COMPONENTES DE SISTEMA OBD-2 [TOMADA DE:(SOCASI 2016)]* 17

FIGURA 3 MODOS DE MEDICIÓN [TOMADA DE:(TENORIO CÓRDOVA, CORONEL MAGALLANES 2020)] 20

FIGURA 4 PARTES DEL SISTEMA DAQ [FUENTE:(AGUERO 2017)] 20

FIGURA 5 PLACAS ARDUINO [TOMADO DE:(ARDUINO | ARDUINO.CL 2019)] 25

FIGURA 6 DIAGRAMA MAPA CONCEPTUAL (FUENTE: ELABORACIÓN PROPIA). 32

FIGURA 7 DIAGRAMA CLASE DISEÑO (FUENTE: ELABORACIÓN PROPIA). 44

FIGURA 8 DISEÑO DE BASE DATOS, DIAGRAMA ENTIDAD-RELACIÓN (FUENTE: ELABORACIÓN PROPIA). 47

FIGURA 9 DIAGRAMA DE COMPONENTES DEL SISTEMA (FUENTE: ELABORACIÓN PROPIA). 51

IGURA 10 DIAGRAMA DE DESPLIEGUE (FUENTE: ELABORACIÓN PROPIA). 52

FIGURA 11 INTERFAZ GRÁFICA INICIO (FUENTE: ELABORACIÓN PROPIA). 53

FIGURA 12 PRUEBA UNITARIA, CAMINO BÁSICO (FUENTE: ELABORACIÓN PROPIA) 57

FIGURA 13 PRUEBA DE RESISTENCIA TIEMPO DE RESPUESTA POR SEGUNDO, MÉTODO GUARDAR HISTORIAL (FUENTE: TEST REPORT FOR LOCUS) 61

FIGURA 14 RESULTADO DE LAS PRUEBAS DE RENDIMIENTO (FUENTE: ELABORACIÓN PROPIA) 62

FIGURA 15 GRAFICO DE EMISIONES DE GASES DE EFECTO INVERNADERO DESDE 1960-2019 [TOMADO DE:(VILLATORO 2019)] 75

FIGURA 16 MAPA POR PAÍSES DE EMISIÓN DE CO_2(%) EN TRASPORTE EN 2019 [TOMADO DE:(HTTPS:// DATOSMACRO.COM) 75

FIGURA 17 EMISIONES DE CO_2 EN CUBA 1960-2020 (KILOTONELADAS) [TOMADO DE : HTTPS://WWW.INDEXMUNDI.COM/ES/DATOS/CUBA/INDICADOR/EN.ATM.CO_2E.SF.KT)] 76

FIGURA 18 INTERFAZ GRÁFICA, ESTADÍSTICA (FUENTE: ELABORACIÓN PROPIA). 80

FIGURA 19 INTERFAZ GRÁFICA, DIAGNOSTICO (FUENTE: ELABORACIÓN PROPIA). 81

FIGURA 20 INTERFAZ GRÁFICA, GRÁFICA SENSORES (FUENTE: ELABORACIÓN PROPIA). 81

FIGURA 21 INTERFAZ GRÁFICA, ESTADÍSTICAS (FUENTE: ELABORACIÓN PROPIA) 82

INTRODUCCIÓN

Durante los 70's y 80's aparecieron componentes electrónicos de control, en los automóviles para controlar y conocer las emisiones de CO_2 emitidas, mediante el dispositivo OBD (Diagnostico abordo u *On Board Diagnostic*), con el pasar del tiempo este sistema se volvió más sofisticado, hasta que en los 90's surgió el sistema OBD2 (*On Board Diagnostic - Second Generation*). Es una de las innovaciones que ha revolucionado la industria automotriz su objetivo principal es el de detectar fallas que se encuentren en el sistema de inyección lo más efectivamente para contrarrestar las emisiones provocadas en la combustión y el control del estado técnico del vehículo mediante la adquisición de los datos de los sensores del mismo (Córdova, Magallanes 2020).

El transporte es el único sector en el que las emisiones de gases de efecto invernadero han aumentado en las últimas tres décadas, con un incremento del 33,5% entre 1990 y 2019. Los vehículos que utilizan combustibles fósiles son un tema particularmente tratado por ser el producto de la combustión producida en los motores al generar la energía para producir el movimiento de los vehículos. Estos motores de combustión interna, que la mayoría emplea como combustible gasolina o diésel, emiten al ambiente diversos gases que alteran la composición natural y contribuyen, además, a reacciones químicas o físicas que deterioran la calidad del aire en el ambiente. (Emissions Gap Report 2020 2021).

En los Anexos, Figura 15, se representa los índices de emisión de CO_2 por los países mayores emisores, desde 1990 hasta 2019. Se observa la alta emisión de gases de efecto invernadero por CO_2 han ido en aumento en los últimos años, según (Emissions Gap Report 2020 2021) "Hasta la fecha, las medidas de mitigación han sido insuficientes, algo que ha complicado considerablemente el logro de los objetivos del Acuerdo de París", a pesar del desarrollo de sistema OBD2, para el control de la emisión de los gases mediante este dispositivo.

Se debe tener en cuenta que la adquisición del sistema OBD2 y su utilización no siempre son de gran comodidad para un cliente y en el mundo la mayor parte de los talleres de reparación y mantenimiento cobran en muchas ocasiones cuotas considerables por conectar el vehículo al escáner y diagnosticar problemas del sistema con OBD2 y que los únicos que manejan los dispositivos para este chequeo son los mecánicos y técnicos automotores influyendo errores del factor humano (Córdova, Magallanes 2020).

Esto ha servido de incentivo para una alternativa en dispositivos de escaneo, más económicos y fácil de usar, por lo que a base de investigación se ha creado y presentado una versión muy práctica para que cualquier persona pueda acceder a las señales del OBD2 y utilizarlo para sus propias pruebas y reparaciones. Surgiendo software que permite a un dispositivo, a través de placas Arduino y Raspberry Pi realizar la función de escáner automotriz para el diagnóstico de vehículos (Córdova, Magallanes 2020).

En Cuba, se tiene como objetivo la reducción de la emisión de CO_2 de los vehículos, con un índice de 4.13 % según (www.datos.bancomundial.org), en aras de contribuir al cuidado del medio ambiente, pero dado las afectaciones debido al bloqueo económico y financiero existente, la adquisición de sistemas para el diagnóstico de vehículos, es muy difícil y costosa, dado consigo la poca disponibilidad de dispositivos con sistema para los diagnósticos de vehículos. Esta situación trae consigo que solamente exista un método de diagnóstico en el país, el llamado Somatón (Revisión Técnica Automotriz) por la empresa ERTA, los cuales se ejecutan cada determinado intervalo de tiempo.

En aras de apoyar el control del mantenimiento de los vehículos de la empresa XETID, la cual posee cifras de gastos en mantenimientos elevadas, en los últimos 3 años, superando los $ 100,000,000 cup al año, los cuales son provocados a la falta de diagnóstico del estado de sus vehículos, dado por la tardanza entre las inspecciones provocando gastos por manteamiento elevados, gastos por traslado de sus vehículos hacia los turnos, en los cuales ocurren en ocasiones viajes sin poder realizarse el diagnóstico, y en el pago de turnos en el Somatón. La empresa poseen altos números de emisión de CO_2 proveniente de sus vehículos, con una flota de más de 30 vehículos, los cuales emiten diariamente 430 560 g de CO_2 siendo cifras elevadas.

Dicha empresa posee un dispositivo Arduino, pero no el software que realizar estos diagnósticos que permitan revertir esta situación. Partiendo de la situación anterior se plantea el siguiente **problema de la investigación:** ¿Cómo adquirir los datos provenientes de la Unidad de Control Electrónico (ECU) de los vehículos mediante dispositivo OBD2?

Se define como **objeto de estudio** los procesos para el diagnóstico de vehículos, y como **campo de acción** los procesos para la adquisición de datos en vehículos. Se define como **objetivo general:** Implementar una aplicación informática para la adquisición de datos en vehículos mediante el dispositivo OBD2.

Luego se derivan los siguientes **objetivos específicos**:

✓ Elaborar el marco teórico de la investigación mediante.

✓ Modelar la propuesta de solución para un sistema de adquisición de datos mediante los artefactos ingenieriles que sugiere la metodología de desarrollo de software seleccionada.

✓ Implementar el sistema de adquisición de datos.

✓ Validar el sistema en función de los requisitos planteados.

Los **métodos teóricos** utilizados son:

- Análisis-Sintético: Este método facilitó un estudio a mayor profundidad acerca de la creación de interfaz para la visualización de datos provenientes de ECU mediante el dispositivo OBD2 y un Arduino.

- Histórico-Lógico: Permite realizar un estudio exhaustivo de los antecedentes históricos de los sistemas de gestión de variables mediante sistema OBD2, teniendo en cuenta su evolución hasta la actualidad. Esto proporciona que se efectúe una investigación acerca del funcionamiento de estos sistemas y su desarrollo, basándose en datos obtenidos a partir del estudio histórico.

- Modelado: permitió crear abstracciones con el objetivo de explicar la realidad. Es empleado como herramienta para la comprensión del problema a resolver y para la creación de los modelos que permitan el diseño de la solución.

Los métodos empíricos que se aplicarán son los siguientes:

- Entrevista: se entrevistó a al cliente y a los especialistas del área de la mecánica del centro para conocer sus necesidades y principales parámetros que necesitaran conocer.

- Observación: posibilitó estudiar y observar el funcionamiento de sistemas similares, identificándose funcionalidades y características que se tuvieron en cuenta para el desarrollo de la solución.

- Encuesta: se efectuó una encuesta para validar la conformidad con los requerimientos y funcionalidades detectadas, con el cliente y especialistas en el área de la mecánica.

- Revisión documental: permitió la obtención de información relacionada con la temática abordada mediante el asesoramiento a través de diferentes artículos y libros.

En este documento de investigación se desglosa todo el procedimiento y metodología usada para lograr los alcances y metas establecidas, así como las indagaciones realizadas con el fin de desarrollar una aplicación lo más completa y eficiente posible.

El documento estará estructurado en tres capítulos, conformados con la información que se presenta a continuación.

Capítulo 1: Fundamentación Teórica sobre los dispositivos OBD2 y los sistemas de adquisición de datos.

En este capítulo se abordan los conceptos y definiciones asociadas al objeto de investigación, referentes a los procesos para el diagnóstico de vehículos, enfatizando en el campo de acción relacionado con los procesos para la adquisición de datos en vehículos. Complementariamente, se evalúan algunos sistemas homólogos con el objetivo de encontrar similitudes en cuanto a características, herramientas y metodologías, que puedan servir como referentes para la implementación de una aplicación informática de adquisición de datos en vehículos mediante el dispositivo OBD2.

Capítulo 2. Propuesta de solución para una aplicación informática para la adquisición de datos en vehículos mediante el dispositivo OBD2.

Se realiza una descripción de la propuesta de solución y se exponen los requisitos funcionales y no funcionales identificados. Teniendo en cuenta la metodología de desarrollo de software seleccionada, se realizará la planificación y el diseño de la propuesta de solución, y se presentan los artefactos ingenieriles que documentan el proceso de desarrollo del software.

Capítulo 3. Pruebas para una aplicación informática para la adquisición de datos en vehículos mediante el dispositivo OBD2.

Se realiza una descripción de la propuesta de solución y se exponen los requisitos funcionales y no funcionales identificados. Teniendo en cuenta la metodología de desarrollo de software seleccionada, se llevará a cabo la planificación y el diseño de la propuesta de solución, y se presentan los artefactos ingenieriles que documentan el proceso de desarrollo del software.

Capítulo 1: Fundamentación Teórica sobre los dispositivos OBD2 y los sistemas de adquisición de datos.

Con base en este proyecto de creación de una aplicación informática para la adquisición de datos en vehículos mediante el dispositivo OBD2, se realizó una búsqueda sobre temas y estudios antes desarrollados semejantes al proyecto, lo cual permitió obtener un análisis e instrucciones para tener una base y obtener conceptos más específicos.

1.1 Historia del OBD

El sistema de OBD incorporado como estándar en los automóviles actualmente fabricados, nace como una solución para regular la emisión de gases generados por la combustión en los automóviles. La estandarización fue un trabajo entre fabricantes, gobierno y entidades preocupadas por el medio ambiente, la cual llevó varias décadas de investigación para poder obtener un sistema eficiente y de altas prestaciones (Contrera Ramírez 2020). La interfaz estándar del OBD2 no solamente es utilizada por el fabricante para sus funciones avanzadas de diagnóstico, sino también por aquellos que van más allá de lo que la ley exige. La siguiente etapa planeada es el OBD3, en el que los propios automóviles se comunican con las autoridades si se produce un empeoramiento de las emisiones de gases nocivos mientras está en marcha. Si esto sucede, se pedirá a través de una tarjeta indicativa, que se corrijan los defectos (SOLÍS, ISRAEL 2010).

Sistema OBD-1

El OBD no tuvo lugar hasta que los autos estuvieron equipados con controles computarizados. La compañía General Motors implementó una versión primeriza de OBD en algunos de sus vehículos de 1980 (SOLÍS, ISRAEL 2010).

El objetivo del sistema OBD original era promover el aire limpio por medio de asegurar que los componentes del control de emisiones siguieran funcionando. Muchos estados de la unión americana empezaron a incluir una revisión de emisiones en su programa de revisión vehicular de seguridad anual o bienal. Pero, en la práctica, mucho puede suceder al vehículo en un año entre verificaciones, y las pruebas simplificadas de entonces solamente “olían” el escape mientras el vehículo estaba estacionado, no cuando estaba en movimiento (SOLÍS, ISRAEL 2010).

La idea del OBD era tener un vehículo que realizará su propio monitoreo del control de emisiones, todo el tiempo, y más que eso asignar números de código que pudiesen identificar el problema, y finalmente guardar esos códigos de problema en la computadora. Una luz indicadora de problema en

el tablero del vehículo podría indicar al conductor que hay un problema en el sistema de emisiones, y cuando el auto estuviese en el taller, el técnico podría extraer los códigos y saber que partes del sistema debería revisar y reparar (SOLÍS, ISRAEL 2010).

Figura 1 Interfaz OBD1 [Tomada de: (Córdova, Magallanes 2020)]

Sistema OBD2

El sistema OBD2 surge a partir de 1990, teniendo como objetivo monitorear los componentes que afecten el sistema de control de emisiones de gases contaminantes y medir parámetros en tiempo real como: temperaturas, presiones, velocidades, entre otros (Anastasio, Enrique 2013).

El OBD2 no tiene mucha diferencia con el sistema OBD1, ya que constituye un perfeccionamiento de este sistema, lógicamente con ciertas diferencias bien definidas, como factores que entre ellos pueden destacar (Anastasio, Enrique 2013):

Características

Las características del OBD2 según (Córdova, Magallanes 2020) son:

• El sistema OBD2 controla virtualmente todos los sistemas de control de emisiones y componentes que pueden afectar los gases de escape.

• En muchos casos, un mal funcionamiento puede ser detectado antes que las emisiones excedan en 1.5 veces los niveles estándar para emisiones a 50 mil millas o 100 mil millas.

• Si un sistema o componente ocasiona que se supere el umbral máximo de emisiones o no opera dentro de las especificaciones del fabricante, un DTC (*Diagnostic trouble code)* debe ser almacenado y la lámpara MIL deberá encenderse dentro de dos ciclos de conducción.

- ✓ **Componentes del Sistema OBD2**

Los componentes del sistema OBD2 son: la ECU (*Engine Control Unit*) conocida como la computadora del automóvil, los transductores encargados de enviar los datos hacia la ECU, la luz indicadora de fallas (MIL, *Malfunction Indicator Light*) ubicado en el tablero, y el conector de diagnóstico (DLC, *Data Link Connector*) que sirve de interfaz entre la ECU y los dispositivos de diagnóstico automotriz (Socasi 2016).

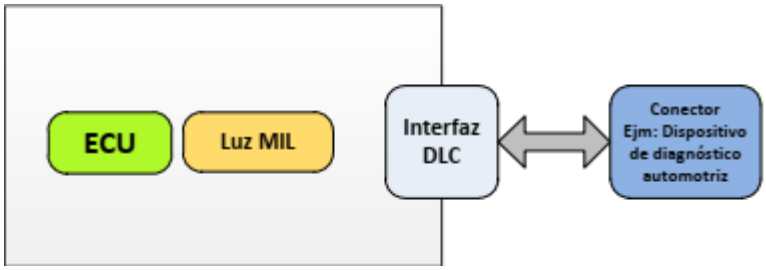


Figura 2 *Componentes de sistema OBD-2 [Tomada de:(Socasi 2016)]*

- ✓ ***Unidad de Control Electrónico (ECU)***

La ECU es la computadora del automóvil y su función principal es obtener y manejar los datos provenientes de todos los transductores del motor del automóvil. Es un dispositivo que se encuentra generalmente debajo del tablero en la parte del conductor (SOLÍS, ISRAEL 2010).

La etapa de entrada se encarga de recibir todas las señales provenientes de los sensores, sean analógicas o digitales. El convertidor Analógico/Digital y el acondicionador de señales forman parte de esta etapa, y tienen la finalidad de digitalizar todas las señales para su posterior procesamiento (Dimaté Cáceres, González Castillo 2013).

La etapa de procesamiento está encargada de procesar los datos mediante la utilización de procesadores, memorias, unidades aritméticas y señales de sincronización (Dimaté Cáceres, González Castillo 2013).

Las etapas de salida se constituyen por las señales de control de los actuadores como: la chispa de ignición, el pulso de control de inyectores, el encendido del ventilador, entre otras (Dimaté Cáceres, González Castillo 2013).

- ✓ ***Sensores del automóvil***

Los sensores son dispositivos encargados de monitorear de forma continua el funcionamiento y operación del motor del automóvil, mediante la conversión de una magnitud física (temperatura, revoluciones por minuto (RPM), etc.) o química (calidad de aire, nivel de oxígeno, etc.) en magnitud eléctrica. Una característica muy importante de los sensores es su exactitud en las mediciones, así como su rapidez de respuesta (Simbaña Coyago 2015).

Los sensores son importantes porque permiten registrar en forma precisa los estados reales del motor de un automóvil en funcionamiento, tales como: RPM del motor, temperatura del líquido refrigerante del motor, presión absoluta del colector de admisión, presión barométrica, temperatura del aire de admisión, posición de acelerador, velocidad del automóvil (Iván Cisneros Rodríguez, s. f.). Dichas medidas hacen que la ECU pueda determinar la cantidad de combustible, el punto de ignición y otros parámetros necesarios para que el automóvil se encuentre en óptimas condiciones (Simbaña Coyago 2015).

- ✓ **RPM del Motor:**

Las Revoluciones por minuto (RPM) indican el número de vueltas completas que desarrolla el cigüeñal; mientras más altas sean las RPM, mayor es la potencia del motor. Además, las RPM ayudan al conductor a saber cuándo se debe realizar un cambio de marcha, mientras el automóvil se encuentra en movimiento (Simbaña Coyago 2015).

- ✓ **Temperatura del líquido refrigerante del motor:**

El líquido refrigerante cumple un papel importante en el automóvil, es el que absorbe el calor y mantiene al motor a una temperatura adecuada de funcionamiento. Si la temperatura del líquido refrigerante se encuentra entre 85 °C y 95 °C, el automóvil, la medida de temperatura del líquido refrigerante se obtiene a través del sensor ECT (*Engine Coolant Temperature*) (Tenorio Córdova, Coronel Magallanes 2020).

- ✓ **Presión absoluta del colector de admisión:**

La presión absoluta del colector de admisión permite a la ECU controlar el tiempo de ignición y ajustar la mezcla de aire-combustible en diferentes condiciones de altitud sobre el nivel del mar. Si la altura es mayor, existe menor cantidad de oxígeno y a consecuencia de eso la combustión del motor es más pobre. La presión absoluta del colector de admisión se obtiene a través del sensor MAP (*Mainfold Absolute Pressure*), su funcionamiento depende de un chip de silicona ubicado dentro del

sensor, que dependiendo de la presión que ingresa por el tubo de admisión, varía su resistencia y a la vez el voltaje de salida del sensor hacia la ECU (Simbaña Coyago 2015).

- ✓ **Posición de acelerador:**

La posición del acelerador indica el estado en el que el pedal se encuentra presionado. Esta medida es adquirida a través del sensor TPS (*Throttle Position Sensor*), permitiendo tener un control adecuado de la mezcla aire-combustible y de la cantidad de combustible que necesitan los inyectores. El funcionamiento del sensor TPS se basa en la variación de la resistencia provocada por el desplazamiento de un brazo cursor conectado al eje de la mariposa. El desplazamiento del brazo cursor hace que la resistencia varíe, permitiendo conocer la posición angular de la válvula de mariposa (Tenorio Córdova, Coronel Magallanes 2020).

- ✓ **Velocidad del automóvil:**

El sensor de velocidad del automóvil (VSS, *Vehicle Speed Sensor*) es un captador magnético que se encarga de enviar a la ECU la velocidad con la que recorre el automóvil, realizando el enriquecimiento de combustible durante la aceleración o el corte de combustible durante la desaceleración (Tenorio Córdova, Coronel Magallanes 2020).

- ✓ **Presión barométrica:**

El sensor de presión barométrica se encarga de medir la presión atmosférica que varía por los cambios del clima y altitud. El funcionamiento del sensor de presión barométrica es similar al del sensor MAP, es decir la variación del voltaje de salida del sensor depende de la presión obtenida. La medida de presión barométrica permite verificar ciertos parámetros en el automóvil tales como: rendimiento en el encendido, emisión de humo negro y consumo de combustible (Simbaña Coyago 2015).

- **Modos de Medición**

El sistema OBD2 emplea nueve modos de medición, en el cual cada uno de ellos permite el acceso a los datos de la ECU del vehículo. Para poder tener acceso es necesaria la utilización de códigos PID (Parámetros de identificación). Cada PID está relacionado con una medida específica de los modos 1 y 2 del sistema OBD2. Por ejemplo, si queremos obtener el dato en tiempo real de las revoluciones del automóvil, se debe ingresar al modo 1 y utilizar el PID 0E (SOLÍS, ISRAEL 2010).

Figura 3 Modos de medición [Tomada de:(Tenorio Córdova, Coronel Magallanes 2020)]

1.2 Sistema de adquisición de datos

La adquisición de datos es un proceso mediante el cual fenómenos físicos del mundo real (sistema analógico) son transformados en señales eléctricas. Estas señales son medidas y convertidas en formato digital (conversión analógica-digital) para su procesamiento, análisis y almacenamiento en una computadora. Para ello se utiliza un módulo de digitalización o tarjeta de Adquisición de Datos (DAQ). Un sistema de adquisición de datos se compone básicamente de (Aguero 2017):

- ✓ Sensores y transductores
- ✓ Acondicionamiento de la señal
- ✓ Hardware y software para la adquisición de datos, PC (sistema de operación)

Figura 4 Partes del sistema DAQ [Fuente:(Aguero 2017)]

Transductores y sensores: Los transductores y sensores son la interface entre el mundo real y el sistema de adquisición. Estos convierten un fenómeno físico en señales eléctricas. El transductor es un elemento de censado que responde directamente a la cantidad física que se quiere medir (el

transductor es referido frecuentemente como un sensor). Como todos los instrumentos de medición, estos tienen un rango de trabajo.

Acondicionamiento de la señal: Las señales de los sensores o del mundo exterior pueden ser ruidosas o demasiado peligrosas para medirse directamente. El circuito de acondicionamiento de señales manipula una señal convirtiéndola en una señal apropiada para la entrada de la tarjeta de adquisición de datos. Este circuito puede incluir amplificación, atenuación, filtrado y aislamiento.

Hardware – software para la adquisición de datos (DAQ): En esta etapa la señal analógica medida se convierte a formato digital (usando un conversor analógico-digital: conversor A/D) y luego los datos se transfieren a una computadora (mediante un software) para su almacenamiento y análisis. Un conversor A/D es un chip que proporciona una representación digital de una señal analógica en un instante de tiempo. En la práctica, las señales analógicas varían continuamente con el tiempo y un conversor A/D realiza "muestras" periódicas de la señal a una razón predefinida. Estas muestras son transferidas a una PC donde la señal original es reconstruida.

En este último paso las tarjetas de adquisición de datos son las que se encargan de convertir las señales analógicas a digitales (conversor A/D) y de la comunicación con el ordenador.
Los conversores A/D presentan varias ventajas:

- ✓ Flexibilidad de procesamiento
- ✓ Posibilidad de realizar las tareas en tiempo real o en análisis posteriores (a fin de analizar los posibles errores)
- ✓ Gran capacidad de almacenamiento
- ✓ Rápido acceso a la información y toma de decisión

Un sistema DAQ básico sería un Arduino que recoge los datos, los procesa y los guarda en una tarjeta SD o un PC conectado, pero con Arduino podemos ir más allá y crear una red de sistemas DAQ interconectados que procesan los datos que capturan y los mandan a una base de datos o repositorio único o distribuido (Crespo 2016).

1.3 Aplicaciones escritorio

Las **aplicaciones de escritorio** son el conjunto de programas o herramientas que tenemos instalados en nuestros ordenadores de sobremesa o portátiles y que únicamente podemos usar en dichos dispositivos. No los podemos trasladar a otro distinto, por lo que posee esta limitación (School 2022).

A la hora de desarrollar una **aplicación de escritorio**, un desarrollador se enfrenta al dilema de qué lenguaje utilizar para que la aplicación se convierta en lo más multiplataforma posible. Y es que una de las maneras de tratar la portabilidad que comentábamos antes es a través de estos lenguajes (School 2022).

Y es que están basados en la máquina virtual que con su propio código puede ejecutar una aplicación software sobre las distintas plataformas existentes. Los lenguajes más conocidos y usados son **Java y Python**, pero como vamos a ver existen muchos más (School 2022).

- ✓ En Windows tenemos Visual C++ y Visual Basic.
- ✓ C/C++ con Qt o GTK.
- ✓ Python con PySide, PyQt, GTK.
- ✓ Java con AWT o Swing.
- ✓ En Mac tenemos Objetive-C/Swift con Cocoa.
- ✓ En Linux tenemos C/C++ con Qt o GTK.

1.4 Estudio de homólogos

A continuación, se muestra en la siguiente tabla una recopilación de aplicaciones y herramientas existentes en el mercado para la adquisición de datos mediante el dispositivo OBD2.

Tabla 1 Estudio de Aplicaciones Homologas (Fuente: Elaboración Propia)

Nombre	Plataforma de despliegue	Funcionabilidades	Interoperabilidad	Limitaciones en cuanto funcionamiento	Código fuente
ScanTool	Aplicación Desktop Windows	Admite opciones de cobertura para 48 fabricantes de automóviles.	Exporta los datos en formato XML y CSV	Bajo licenciamiento	Software privativo
ProScan	Aplicación Desktop Windows	Pruebas del sensor de oxígeno. Generador de informes de diagnóstico.	No es posible	Bajo licenciamiento	Software privativo
PCMSCAN	Aplicación Desktop Windows	Permite ver, crear gráficos, registrar y reproducir datos de diagnóstico en el momento. Es compatible con todos los automóviles	No es posible	Bajo licenciamiento	Software privativo
EOBD Facile	Aplicación Desktop Mac	Ver fallas del motor de códigos y descubrir su significado Supervisar códigos de error	No es posible	Bajo licenciamiento	Software privativo
OBD2 Auto Doctor	Aplicación Desktop Mac	Permite al usuario examinar y restablecer los códigos de alerta.	Posibilidad de enviar estos datos por correo electrónico.	Bajo licenciamiento	Software privativo

A partir del análisis de los sistemas evaluados en la Tabla 1 se tomaron en cuenta algunas características como, plataforma de despliegue, funcionalidades del software, interoperabilidad, limitaciones en cuanto a funcionamiento, y código fuente. Como se puede observar, todos estos software son privativos, por tal razón no es posible la modificación de su código para adecuarlo a las necesidades del cliente, también la adquisición de los mismos es costosa dado las afectaciones del bloqueo, por lo que ninguno de estos sistemas resuelve el problema. Sin embargo, cuentan con funcionalidades que se pueden tener presentes en el desarrollo de una nueva solución, entre las que se destacan:

- ✓ Realizar monitoreo en tiempo real.
- ✓ Obtener los códigos de fallas y su descripción.
- ✓ Registrar el historial de diagnósticos.
- ✓ Graficar los resultados de los sensores.

1.5 Plataforma Arduino

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un Microcontrolador reprogramable y una serie de pines hembra. Estos permiten establecer conexiones entre el Microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables DuPont) (*Arduino | Arduino.cl* 2019).

A lo largo de los años, Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta instrumentos científicos complejos. Una comunidad mundial de creadores (estudiantes, aficionados, artistas, programadores y profesionales) se ha reunido en torno a esta plataforma de código abierto, sus contribuciones se han sumado a una increíble cantidad de conocimiento accesible que puede ser de gran ayuda tanto para principiantes como para expertos (*Arduino | Arduino.cl* 2019).

Arduino nació en el Instituto de Diseño de Interacción Ivrea como una herramienta fácil para la creación rápida de prototipos, dirigida a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y desafíos, diferenciando su oferta de simples placas de 8 bits a productos para aplicaciones IoT*, wearables*, impresión 3D y entornos integrados (*Arduino | Arduino.cl* 2019) .

Figura 5 Placas Arduino [Tomado de:(Arduino | Arduino.cl 2019)]

En la Unidad Básica de Investigación y Desarrollo de la XETID a pesar de la existencia de varios modelos de placas Arduino, se desarrolló un prototipo de dispositivo OBD2 basado específicamente en el Arduino UNO.

1.6 Ambiente de desarrollo

A continuación se caracterizan todas las herramientas y lenguajes de programación utilizados para la elaboración de un nuevo sistema.

Visual Paradigm 16.2

Visual Paradigm es una herramienta *Computer Aided Software Engineering* (CASE, por sus siglas en inglés) que emplea UML como lenguaje de modelado. Las herramientas de Ingeniería de Software Asistida por Computadora se pueden definir como un conjunto de programas y ayudas que dan asistencia a los analistas, desarrolladores e ingenieros de software, durante el ciclo de vida de un software, proporcionándole un aumento en su productividad y logrando un mayor ahorro de tiempo (Ambler 2017).

Visual Paradigm soporta el ciclo de vida de desarrollo de software completo, desde el análisis y diseño hasta la construcción, pruebas e incluso despliegue. Permite representar gráficamente varios

diagramas y facilita la generación de código. Es una herramienta multiplataforma, fácil de instalar y utilizar. A continuación, se listan algunas de sus principales ventajas según el sitio oficial (*Visual Paradigm* 2022):

- ✓ Soporte nativo de UML
- ✓ Diagramas de procesos de negocio
- ✓ Diagramas entidad relación y transformación en tablas de bases de datos
- ✓ Soporte a técnicas de ingeniería inversa
- ✓ Capacidad de convertir un modelo en código

Lenguaje de Modelado (UML) 8.0

Como lenguaje de modelado se seleccionó UML en su versión 8.0, dado que es el lenguaje de modelado de Sistemas de software más conocido y utilizado en la actualidad. Es una consolidación de muchas de las notaciones y conceptos más usados en la metodología orientada a objetos. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios, funciones del sistema, aspectos concretos como expresiones de lenguajes de programación, componentes de software reutilizables y esquemas de bases de datos.

Además, permite especificar y visualizar un sistema. Se puede utilizar para definir un sistema de software, detallar los artefactos, especificar su documentación y construcción. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software, contando con varios tipos de diagramas para llegar a representar los diferentes puntos de vista de un sistema (Jacobson, Booch, Rumbaugh 2006).

IDE Arduino 2.0

El IDE es un conjunto de herramientas de software que permiten a los programadores desarrollar y grabar todo el código necesario para hacer que nuestro Arduino funcione como queramos. El IDE de Arduino nos permite escribir, depurar, editar y grabar nuestro programa (llamados “sketches” en el mundo Arduino) de una manera sumamente sencilla, en gran parte a esto se debe el éxito de Arduino, a su accesibilidad (*Arduino | Arduino.cl* 2019).

Lenguaje Arduino 2.0

El equipo que desarrolló la tarjeta Arduino creó un lenguaje propio para programar estas tarjetas. Este lenguaje está basado en un lenguaje llamado *Wiring*. Este es un framework (Un framework es una especie de plantilla, esquema o estructura conceptual de base tecnológica que nos permite

trabajar de una manera mucho más sencilla) de código abierto para la programación de microcontroladores (Concepción 2020).

Qt Designer 4.6

Es una herramienta de Qt que le proporciona una interfaz de usuario de lo que ve es lo que obtiene (WYSIWYG) para crear GUI para sus aplicaciones PyQt de manera productiva y eficiente. Con esta herramienta, crea GUI arrastrando y soltando objetos QWidget en un formulario vacío. Después de eso, puede organizarlos en una GUI coherente utilizando diferentes administradores de diseño.

Qt Designer también le permite obtener una vista previa de sus GUI usando diferentes estilos y resoluciones, conectar señales y ranuras, crear menús y barras de herramientas, y más.

Qt Designer es independiente de la plataforma y el lenguaje de programación. No produce código en ningún lenguaje de programación en particular, pero crea archivos .ui. Estos archivos son archivos XML con descripciones detalladas de cómo generar GUI basadas en Qt.

Puede traducir el contenido de los archivos .ui a código Python con pyuic 5, que es una herramienta de línea de comandos que viene con PyQt. Entonces puede usar este… código Python en sus aplicaciones GUI. También puede leer archivos .ui directamente y cargar su contenido para generar la GUI asociada (Python).

PySide 6.0

PySide es una biblioteca de Python para crear interfaces gráficas de usuario multiplataforma. Es un enlace de Python al marco Qt. La biblioteca Qt es una de las bibliotecas GUI más poderosas. PySide se implementa como un conjunto de módulos de Python. Actualmente cuenta con 15 módulos. Estos módulos proporcionan potentes herramientas para trabajar con GUI, multimedia, documentos XML, red o bases de datos (*Introduction to PySide toolkit*).

Visual Studio Code 1.66

Es un editor de código multiplataforma, de código abierto y gratis desarrollado por Microsoft. Tiene soporte nativo para diferentes lenguajes de programación como JavaScript, Typescript, CSS, SASS y tecnologías como JSON, HTML, entre otros. Dentro de sus características destaca el soporte IntelliSense, con resaltado de sintaxis, autocompletado y detección de errores. Está equipado con utilidades para la detección de errores en el funcionamiento del código (debug) e integración completa con Git, permitiendo controlar las versiones del proyecto de forma transparente y fácil.

Posee capacidad de incorporar nuevas funcionalidades instalando extensiones desarrolladas por la comunidad o por equipos de desarrollo profesionales (*Visual Studio Code* 2022).

Python 3.8

Es un lenguaje de programación de **alto nivel** que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario "traducirlo" a lenguaje máquina.

Python es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano. Además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites. Con el paso del tiempo, Python ha ido ganando adeptos gracias a su sencillez y a sus amplias posibilidades, sobre todo en los últimos años, ya que facilita trabajar con inteligencia artificial*, big data, machine learning* y *data science*, entre muchos otros campos en auge (*Python: qué es y por qué deberías aprender a utilizarlo*).

SQLite 3.0

SQLite es una biblioteca en lenguaje C que implementa un motor de base de datos SQL pequeño, rápido, autónomo, de alta confiabilidad y con todas las funciones. SQLite es el motor de base de datos más utilizado en el mundo. SQLite está integrado en todos los teléfonos móviles y en la mayoría de las computadoras y viene incluido dentro de innumerables otras aplicaciones que la gente usa todos los días.

El formato de archivo SQLite es estable, multiplataforma y compatible con versiones anteriores y los desarrolladores se comprometen a mantenerlo así hasta el año 2050. Los archivos de base de datos SQLite se usan comúnmente como contenedores para transferir contenido enriquecido entre sistemas y como un formato de archivo a largo plazo para datos. Hay más de 1 billón (1e12) de bases de datos SQLite en uso activo. El código fuente de SQLite es de dominio público y es gratuito para todos para cualquier propósito (*SQLite Home Page*).

1.7 Metodología de desarrollo

A decir de (Pressman 2021), "Una metodología es el conjunto ordenado de pasos a seguir para cumplir un objetivo. Dicho objetivo, en la ingeniería de software, es el desarrollo de software de alta calidad que cumple con las necesidades del cliente dentro de un plan y un presupuesto predecible.

Para esto es necesario proveer un enfoque disciplinario para asignar tareas y responsabilidades dentro del desarrollo del sistema, determinar un camino metódico y sistemático para desarrollar, diseñar y validar una arquitectura y reducir en gran medida los riesgos que representa la construcción de sistemas de software".

Metodología AUP variante UCI

Para lograr la mayor eficiencia en el ciclo de vida de la actividad productiva en la UCI se decidió una variación en la metodología AUP (Rodriguez, Vazquez, Aliaga 2012). La metodología consta de 3 fases:

Inicio: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance de este, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no.

Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se realizan pruebas al producto.

Cierre: en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Para el desarrollo de la propuesta de solución se utiliza el escenario 4 de historia de usuario, debido a que el proyecto fue evaluado y como resultado se obtuvo un negocio bien definido. El cliente está siempre acompañado al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. La duración del proyecto no es muy extensa, y no se debe generar mucha información y el equipo de desarrollo está conformado por una sola persona.

Conclusiones del capítulo

En el capítulo se definen los conceptos esenciales para la investigación relacionados con los procesos de adquisición de datos mediante el dispositivo OBD2, permitiendo elaborar un marco teórico, además los cuales son empleados como nomencladores en la propuesta de solución. El análisis de homólogos permitió establecer la existencia de sistemas para la adquisición de datos mediante OBD2. De igual forma, permitió conocer las principales características y funcionalidades que poseen estos sistemas para tener en cuenta a la hora de elaborar un sistema propio, al determinarse que los existentes no son la mejor opción a utilizar al ser de pago y no satisfacer las necesidades del cliente. Para la propuesta de solución se determina el uso de AUP-UCI como metodología de desarrollo, Visual Paradigm como herramienta para el modelado, Python como lenguaje de programación de lado del servidor, Visual Studio Code como entorno de desarrollo, SQlite como gestor de base de datos, PySide como lenguaje de programación del lado del cliente, QtDesigner, como entorno de desarrollo de interfaz y Arduino como entorno de desarrollo para las Placas Arduino.

Capítulo 2: Planificación y diseño del sistema informático para la adquisición de datos en vehículos mediante el dispositivo OBD2.

En el presente capítulo se describen la propuesta de solución, las tareas de ingeniería realizadas y los productos de trabajo obtenidos durante la ejecución de las disciplinas de Requisitos, Análisis y Diseño. Los principales productos obtenidos en esta fase de desarrollo son diagrama de clase de diseño, los requerimientos funcionales y no funcionales que debe cumplir el sistema GesMot-OBD2, así como las Historias de usuarios correspondientes. Por último, se describe la arquitectura del sistema a desarrollar y los patrones de diseño utilizados.

Propuesta de solución. Objetivos y alcance.

El sistema propuesto como solución es un sistema de desktop, el cual, posea una interfaz agradable, de fácil entendimiento, con colores oscuros. El sistema deberá permitir adquirir los datos del vehículo mediante los datos obtenidos mediante la placa Arduino UNO, la cual será utilizada como simulador de OBD2; estos datos serán almacenados y mostrados tanto en tiempo real, como en un historial, donde se grafiquen los valores de los sensores. El sistema debe ser capaz de describir los códigos de fallas del motor. Se desea que el sistema sea capaz de intercambiar información con una plataforma web encargada de gestionar todo el negocio. Todas las funcionalidades deseadas deben ser cumplidos en su totalidad, debe tener tiempo de respuestas corto. Además, debe ser robusto, tolerante a fallos y errores de los usuarios.

2.1 Mapa conceptual

Son organizadores gráficos que permiten representar el conocimiento, entendido como una serie de conceptos percibidos como regulares en una serie de eventos y objetos, los cuales se conectan con palabras vinculantes para formar proposiciones (García Franco 2020).

Los principales elementos que componen un mapa conceptual son (García Franco 2020):

Concepto: se entiende por concepto la palabra o término que manifiesta una regularidad en los hechos, acontecimientos ideas y/o cualidades.

Palabras de enlace: son palabras que unen los conceptos y señalan los tipos de relación existente entre ellos.

Proposición: se establece a partir de la unión de dos o más conceptos ligados por palabras de enlace en una unidad semántica. Corresponde a la unidad principal del significado.

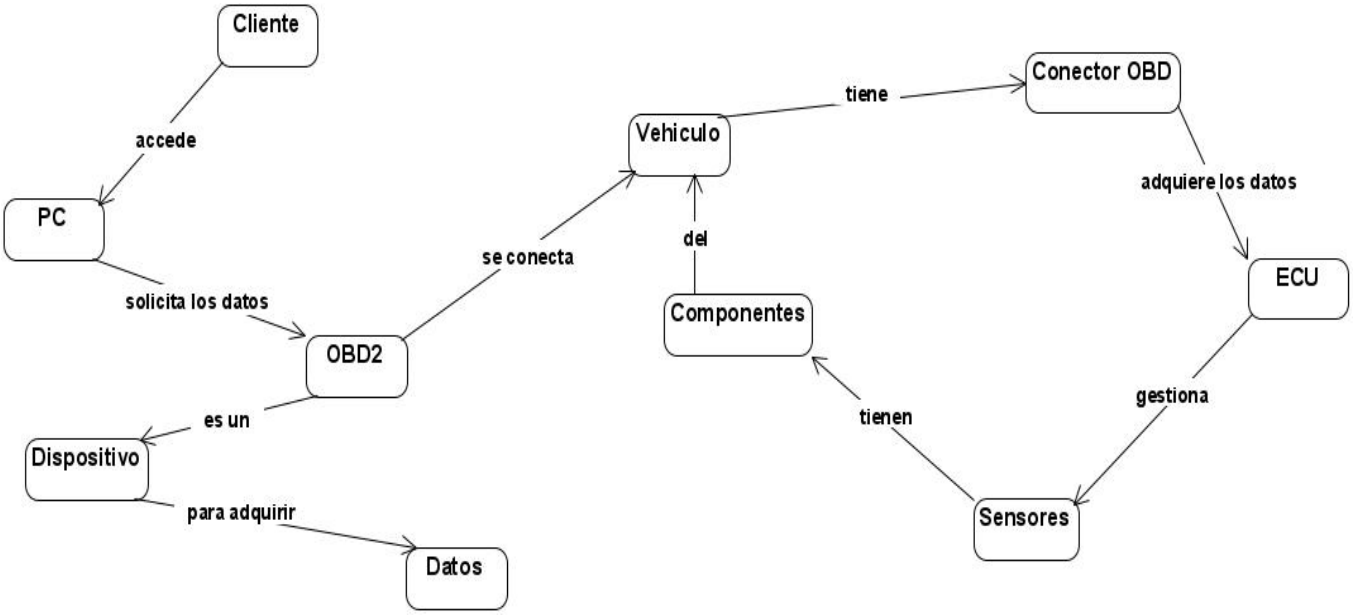


Figura 6 Mapa Conceptual (Fuente: Elaboración propia).

2.2 Levantamiento de Requisitos

Los requisitos de un sistema son la descripción de lo que es sistema debe hacer, los servicios que proporciona y las restricciones de sus operaciones. Dichos requerimientos reflejan las necesidades de los clientes del sistema. El proceso de obtención, análisis, documentación y validación de esos servicios y restricciones se llama ingeniería de requisitos. Los requisitos se clasifican en funcionales (servicios que el sistema debe proporcionar) y no funcionales (restricciones de los servicios o funciones ofrecidos por el sistema) (Sommerville 2020). A continuación, se describe cómo se obtuvieron los requisitos de la propuesta de solución, su especificación, descripción y validación.

Obtención de Requisitos

Las técnicas de identificación de requisitos de software según (Sommerville 2020) son procesos para la obtención de información sobre el sistema requerido y sistemas existentes, permite separar los requerimientos del sistema y los del usuario. Algunas fuentes de información son documentaciones, *stakeholders* y especificaciones de sistemas similares. (Pressman 2021) indica que estas técnicas permiten identificar las necesidades de negocio de los clientes y los usuarios. Son mecanismos que

se utilizan para recolectar la información necesaria en la obtención de los requisitos de una aplicación, permiten investigar aspectos generales para posteriormente ser especificados con un mayor detalle, requieren ser adecuadamente orientadas para cubrir la información que se requiere capturar.

Entrevista

Fuente de información de gran utilidad para obtener información cualitativa como opiniones, o descripciones subjetivas de actividades. Es una técnica muy utilizada, y requiere una mayor preparación y experiencia por parte del analista. La entrevista se puede definir como un “intento sistemático de recoger información de otra persona” a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada. Debe quedar claro que no basta con hacer preguntas para obtener toda la información necesaria. Es muy importante la forma en que se plantea la conversación y la relación que se establece en la entrevista (Guerra 2017).

El Anexo 1 de este documento es la entrevista realizada a los trabajadores de la XETID, la misma permitió obtener información sobre las principales necesidades sobre los datos deseados a obtener con la propuesta de solución.

Estudio de documentación

El estudio de varios tipos de documentación, como manuales y reportes, puede proporcionar al analista información valiosa con respecto a las organizaciones y a sus operaciones. La documentación difícilmente refleja la forma en que realmente se desarrollan las actividades, o donde se encuentra el poder de la toma de decisiones. Sin embargo, puede ser de gran importancia para introducir al analista al dominio de operación y el vocabulario que utiliza (Guerra 2017). El Anexo 1 presenta la guía de estudio de la documentación del proceso de conexión a los dispositivos OBD2, cuáles son los datos que son adquiridos, formas de representarlos. Las técnicas llevadas a cabo para la recopilación de información relacionada con los procesos de los diagnósticos a los vehículos, junto a las opiniones de los trabajadores del centro, esclarecieron aspectos relacionados con la ingeniería de requisitos, específicamente la fase de obtención de requisitos permitiendo al autor de este trabajo de diploma captar las principales necesidades de software y humanas en las que se debe centrar, las mismas serán presentadas en el epígrafe siguiente.

2.3 Especificación de Requisitos

Una especificación de requisitos de software es un documento que se crea cuando debe especificarse una descripción detallada de todos los aspectos del software que se va a elaborar, antes de que el proyecto comience (Pressman 2021). Los requisitos de software de la propuesta de solución se presentan a continuación.

Requisitos funcionales

Para el levantamiento de los requisitos del sistema son empleadas técnicas de captura de requisitos tales como:

- ✓ Tormenta de ideas: se realiza mediante encuentros del equipo de trabajo con el cliente, de modo que tanto desarrolladores como cliente aportaron ideas que facilitaron la obtención de los requisitos funcionales del sistema.
- ✓ Entrevistas: desarrolladas a través de encuentros personalizados con el cliente y con una lista de preguntas previamente elaboradas para poder establecer los objetivos del sistema, qué es lo que debe lograr y de qué forma satisfacer sus necesidades.

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera que este debe reaccionar a entradas particulares y cómo se debe comportar en situaciones particulares (Sommerville 2011).

RF-1 Seleccionar puerto de conexión: la aplicación debe permitir escoger el puerto para la conexión con el dispositivo OBD2.

RF-2 Conectar: la aplicación debe permitir conectarse al dispositivo OBD2.

RF-3 Desconectar: la aplicación debe permitir al dispositivo OBD2.

RF-4 Iniciar adquisición de datos: la aplicación debe permitir iniciar la adquisición de los datos del dispositivo OBD2.

RF-5 Finalizar adquisición de datos: la aplicación debe permitir terminar el proceso de adquisición de datos.

RF-6 Salvar datos adquiridos: la aplicación debe permitir almacenar los datos en una base datos.

RF-7 Visualizar datos adquiridos: la aplicación debe mostrar los datos adquiridos.

RF-8 Visualizar historial de datos adquiridos: la aplicación debe mostrar el historial de datos adquiridos.

RF-9 Visualizar gráfica del historial de datos adquiridos: la aplicación debe mostrar una gráfica con el historial de los datos.

RF-10 Visualizar datos en tiempo real del vehículo: la aplicación debe mostrar una interfaz con los valores en tiempo real de Rpm, Velocidad, Batería y Temperatura del Motor.

Requisitos no funcionales

Los requisitos no funcionales describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Los requerimientos no funcionales incluyen restricciones como el tiempo de respuesta (desempeño), la precisión, recursos consumidos, seguridad, etcétera (Pressman 2021).

Se trata de requisitos que no se refieren directamente a las funciones específicas suministradas por el sistema (características de usuario). Alternativamente, definen restricciones del sistema tales como la capacidad de los dispositivos de entrada/salida y la representación de los datos utilizados en la interfaz del sistema (Pressman 2021).

Los requerimientos no funcionales son los que especifican criterios para evaluar la operación de un servicio de tecnología de información y especifican los criterios que debe cumplir para que sea adecuado para su uso (Pressman 2021).

La ISO/IEC 25010 define el modelo, para evaluar la calidad del producto de software, el cual define ciertas características, Sub-características y métricas, que al ser aplicadas en un producto de software garantizan la calidad del mismo (*ISO 25010*). En el presente estudio estas características se toman como métricas para evaluar el sistema.

Tomando en cuenta el trabajo que se desarrollará, se definen lo siguiente:

Nivel de importancia

En la tabla 2 se especifica que representa cada nivel de importancia aplicados a las características y Sub-características a evaluarse:

Tabla 2 Nivel de importancia (Fuente: Elaboración Propia)

Nivel de importancia	Nomenclatura	Descripción
Alta	A	El nivel de importancia de la característica y sub característica obliga a realizar las mediciones.

Media	M	El nivel de importancia de la característica y sub característica indica que se sujeta a criterio del evaluador.
Baja	B	El nivel de importancia de la característica y sub característica indica que no es necesaria la medición.

Ponderación de características y Sub-características de calidad

Con el objetivo de señalar los resultados cuantitativos calculados en la evaluación del producto software, se va a establecer una ponderación a las características y Sub-características de calidad externa y en uso, dependiendo del nivel de importancia que se asignó a cada una. Se debe tomar en cuenta que las ponderaciones concedidas son de criterio del/a evaluador/a y depende también del tipo de software que se está evaluando; adicionalmente, las ponderaciones a asignarse serán únicamente para las características y Sub-características de calidad que son aplicadas, y su sumatoria debe ser de 100% (Elizabeth 2018).

Definición de métricas a evaluar

A continuación, se detalla las métricas que se van a evaluar en las aplicaciones, la importancia que se le dio a cada una, y el criterio personal por el cual se evaluará cada métrica. Primero desde la tabla 3 hasta la tabla 10, se tiene como métricas que medirán la calidad externa de las tres aplicaciones, las características y Sub-características del modelo de calidad externa para la evaluación de productos software, definidas en la ISO/IEC 25010:

Característica 1 (C1P): Adecuación funcional

Nivel de importancia C1P: Alta

Criterio para evaluar C1P: Es necesario verificar que el aplicativo cumpla con todas las funciones para las cuales fue desarrollado.

Tabla 3 Sub características de la Característica 1 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar
Completitud funcional	A	El sistema debe cumplir con las especificaciones del usuario final

Corrección funcional	A	El sistema debe proveer la mayor cantidad de resultados correctos, con una gran precisión.

Característica 2 (C2P): Fiabilidad

Nivel de importancia C2P: Alta

Criterio para evaluar C2P: El aplicativo debe mantener el mismo nivel de respuesta a las peticiones del usuario en cuanto a tiempo y eficiencia.

Tabla 4 Sub características de la Característica 2 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar
Madurez	M	c
Tolerancia a fallos	A	El principal componente de uso de esta aplicación en el Arduino y la API-REST, los cuales, en caso de fallo, no se podrá realizar las funciones.

Característica 3 (C3P): Eficiencia y desempeño

Nivel de importancia C3P: Media

Criterio para evaluar C3P: Se necesita medir el desempeño del aplicativo en relación con la cantidad de recursos utilizados.

Tabla 5 Sub características de la Característica 3 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar

Comportamiento temporal	M	Los tiempos de respuestas de la aplicación deben estar comprendidos en no más de 3 segundos.
Utilización de recursos	M	La aplicación no debe colapsar debido a la falta de recursos el dispositivo

Característica 4 (C4P): Usabilidad

Nivel de importancia C4P: Media

Criterio para evaluar C4P: Es indispensable que la aplicación sea de entendible por el usuario final.

Tabla 6 Sub características de la Característica 4 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar
Capacidad de reconocer su adecuación	A	La aplicación debe cumplir todos los requerimientos del usuario final y así lograr su satisfacción y uso por parte del mismo.
Capacidad de ser usado	M	El usuario debe familiarizarse con todas las funcionalidades de la aplicación con la finalidad de realizar un uso adecuado y eficiente de la misma.
Capacidad de aprendizaje	A	La aplicación debe tener un flujo entre interfaces intuitivo y bastante sencillo de entender.
Protección contra errores del usuario	M	La aplicación debe ser capaz de evitar posibles errores del usuario
Estética de la Interfaz del usuario	M	La interfaz gráfica de la aplicación no debe interferir en las funcionalidades de la misma.
Accesibilidad	B	Personas con discapacidad visual no pueden utilizar la aplicación.

Característica 5 (C5P): Seguridad

Nivel de importancia C5P: Alta

Criterio para evaluar C5P: Es importante la protección a los datos recibidos y procesados.

Tabla 7 Sub características de la Característica 5 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar
Confidencialidad	A	La aplicación debe ser capaz de mantener los datos de forma confidencial
Integridad	M	La integridad de los datos procesados debe ser una prioridad

Característica 6 (C6P): Compatibilidad

Nivel de importancia C6P: Media

Criterio para evaluar C6P: Se requiere que una aplicación cumpla con los estándares de compatibilidad que dicta la ISO 25010.

Tabla 8 Sub características de la Característica 6 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar
Co – existencia	A	La aplicación nunca debe ser objeto de fallo de otra aplicación que quiera utilizar los mismos recursos.
Interoperabilidad	A	La aplicación debe ser capaz de interactuar mediante micro-servicios con otros sistemas.

Característica 7 (C7P): Mantenibilidad

Nivel de importancia C7P: Media

Criterio para evaluar C7P: Al realizar un cambio o actualización en la aplicación, el esfuerzo del desarrollador debe ser mínimo.

Tabla 9 Sub características de la Característica 7 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar
Capacidad de ser analizado	M	Al realizar una modificación en la aplicación, cualquier falla que se presente en el resto del mismo debe ser fácil de identificar y solucionar
Capacidad de ser modificado	M	Al realizar un cambio o actualización en una parte del aplicativo no debe implicar daños o fallas en el rendimiento del mismo.
Capacidad de ser probado	M	Permitir la realización de pruebas

Característica 8 (C8P): Portabilidad

Nivel de importancia C8P: Media

Criterio para evaluar C8P: La aplicación al instalarse en diferentes dispositivos debe funcionar de la misma manera.

Tabla 10 Sub características de la Característica 8 del Modelo de Calidad Externa, especificando importancia y criterio de evaluación (Fuente: Elaboración propia)

Sub-características	Nivel de Importancia	Criterio para evaluar

Capacidad de ser Instalado	M	La aplicación debe permitir ser instalada en todos los dispositivos, sin importar el sistema operativo

2.4 Descripción de requisitos de software

En el escenario 4 para la obtención de requisitos de la metodología AUP-UCI se emplean las historias de usuarios (HU) como mecanismos de descripción de los requisitos funcionales. Las historias de usuarios son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan estimar el tiempo de desarrollo e implementarlas sin dificultad (Joskowicz 2008). A continuación, se muestran algunas de las historias de usuario de diferentes requisitos de la propuesta de solución con diferentes niveles de complejidad, las demás historias de usuario están en el Anexo 2.

Tabla 11 HU Gestionar Conexión (Fuente: Elaboración propia).

Historia de usuario	
Número: 1	**Número del requisito:** Gestionar Conexión
Programador: Yandry R García Pérez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 96 h
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 96 h
Descripción: ● **Objetivo:** Gestionar la conexión al dispositivo OBD2, para poder adquirir los datos. ● **Precondiciones:** Se debe tener conectado el dispositivo al computador. ● **Acciones a realizar:** **Seleccionar puerto de conexión:** Seleccionar puerto en el **selector** **Conectar:**	

Al dar clic en el **botón conectar** se iniciará la conexión con el dispositivo y se mostrará un icono de conexión en caso de realizarse. En caso de no conectar se mostrará el icono sin conexión y se notificará. **Desconectar:** Al dar clic en el **botón desconectar** se detendrá la conexión con el dispositivo.
Observación:
Prototipo de interfaz: 

2.5 Diseño de la propuesta de solución

En esta fase de desarrollo los requisitos pueden ser refinados y estructurados para conseguir una mejor comprensión de los mismos con una descripción que sea fácil de mantener y soporte la estructura del sistema. En esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales (Rodríguez Sánchez 2015). El diseño de software agrupa el conjunto de principios, conceptos y prácticas que llevan al desarrollo de un sistema o producto de calidad, permite modelar el sistema que se va a construir, proporcionando detalles de la arquitectura del software, estructuras de datos, interfaces y componentes que se necesitan para implementar el sistema (Pressman 2021). En los siguientes sub-epígrafes se hace una definición de los elementos del análisis y diseño considerados para la implementación de la propuesta de solución.

Patrón arquitectónico

Los patrones arquitectónicos o patrones de arquitectura ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Reflejan una descripción de los elementos y el tipo de relación que tienen, seguido de un conjunto de restricciones sobre cómo pueden ser usados (Pressman 2021).

Patrón Modelo-Vista-Controlador (MVC)

Para la organización del proyecto se seleccionó el patrón MVC. Este estilo de arquitectura separa los datos de la aplicación, interfaz de usuario y lógica de control en tres componentes distintos (*Qué es MVC* sin fecha):

Modelo: contiene una representación de los datos que maneja el sistema, su lógica de negocio y sus mecanismos de persistencia.

Vista o interfaz de usuario: compone la información que se envía al cliente y los mecanismos de interacción con este.

Controlador: actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Esta “separación de preocupaciones” proporciona una mejor división del trabajo y una mejora de mantenimiento.

Diagrama de clase del diseño (DCD)

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación, es un conjunto de definiciones de entidades software más que de conceptos del mundo real. Las clases del diagrama pueden tener relaciones entre ellas, tales como relaciones de dependencia, asociativas y herencia (Larman 1999).

A continuación, se muestra el DCD del requisito funcional Salvar datos adquiridos. Los DCD correspondientes a los requisitos funcionales restantes se muestran en el Anexo 3.

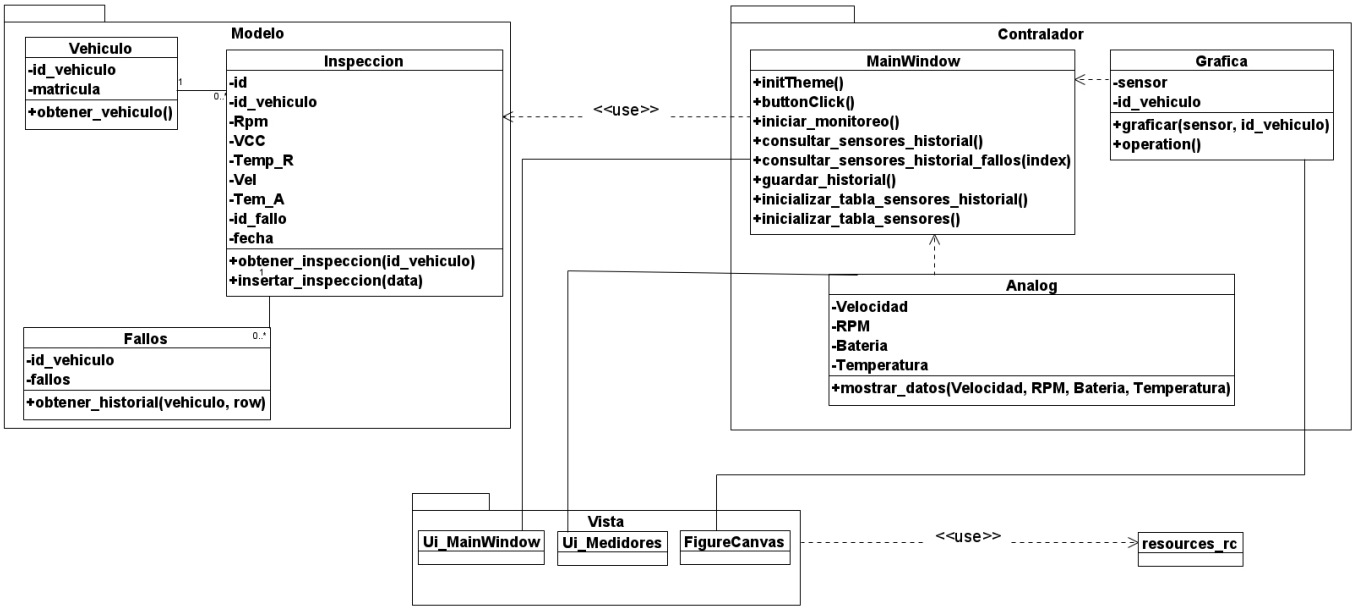


Figura 7 Diagrama clase diseño (Fuente: Elaboración propia).

Patrones de diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico y entre fuerzas que afectan la manera en la que se aplica y en la que se utiliza dicho patrón. El objetivo de cada patrón es proporcionar una descripción que permita a un diseñador determinar si el patrón es aplicable al trabajo en cuestión, si puede volverse a usar y si sirve como guía para desarrollar un patrón distinto en función o estructura (Pressman 2021). El patrón es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos; en teoría, indica la manera de usarlo en circunstancias diversas (Larman 1999).

En resumen, un patrón de diseño es una estructura de diseño que describe un problema y su solución práctica, es un estándar que se utiliza al diseñar sistemas que resuelven determinados problemas comunes y que ayuda a un mejor entendimiento del código por parte de otros programadores.

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa *General Responsibility Assignment Software Patterns* (patrones generales de software para asignar

responsabilidades). El nombre se eligió para indicar la importancia de captar (*grasping*) estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Larman 1999).

Experto: Experto es un patrón que se usa más que cualquier otro al asignar responsabilidades; Es un principio básico que suele utilizarse en el diseño orientado a objetos. Con él no se pretende designar una idea oscura ni extraña; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. Ofrece una analogía con el mundo real. Acostumbramos asignar responsabilidad a individuos que disponen de la información necesaria para llevar a cabo una tarea (Larman 1999), se pone en práctica en las clases *Inspeccion* que es el que posee toda la información necesaria para mostrar y almacenar los datos adquiridos.

Bajo acoplamiento: El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras. El Bajo Acoplamiento es un principio que debemos recordar durante las decisiones de diseño: es la meta principal que es preciso tener presente siempre. Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño (Larman 1999), el uso de este patrón en el diseño facilita la existencia sólo de las relaciones entre clases necesarias, de forma tal que pequeños cambios tengan la mínima repercusión en otras clases. También permite que las clases sean fáciles de entender por separado y de reutilizar.

En la propuesta de solución se utilizó este patrón en clases como *Analog* la cual se relacionan con la menor cantidad de clases posibles para el cumplimiento de sus funciones.

Alta Cohesión: Nos dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.

La clase *Inspeccion* está altamente cohesionada al ser responsable del envío en tiempo real de los datos obtenidos de las inspecciones.

Controlador: Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema (evento de alto nivel generado por un actor externo). Define además el método de su operación (Larman 1999), permite que a través de un archivo se procesen todas las peticiones de manipulación por analizar a través de un objeto de controlador único. En la propuesta de solución se utilizó este método en las clases que realizan solicitudes del usuario, en la propuesta se utiliza en la clase *MainWindow*, la cual va a controlar las funciones sobre la interfaz principal.

Creador: La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento (Larman 1999), se refleja en las clases controladoras ubicadas en el directorio ./controller/, donde se encuentran las acciones definidas para las operaciones lógicas del negocio referentes a las entidades y se ejecutan cada una de ellas.

Patrones GOF

Hay varias categorías de patrones de diseño utilizados en la programación orientada a objetos, según el tipo de problema que abordan y/o los tipos de soluciones que nos ayuden a construir. En su libro, Gang of Four presenta 23 patrones de diseño, divididos en tres categorías: creacional (tratan diferentes aspectos de la creación de objetos. Su objetivo es proporcionar mejores alternativas para situaciones en las que la creación directa de objetos no es conveniente), estructural (propone una forma de componer objetos para crear nuevas funcionalidades) y de comportamiento (se ocupan de los algoritmos y la asignación de responsabilidades entre objetos) (Ayeva, Kasampalis 2018).

Fábrica Abstracta (*Abstract factory*): El patrón de Abstract factory (creacional) se usa normalmente cuando tenemos múltiples posibles implementaciones de un sistema que dependen de algún problema de configuración o plataforma. El código de llamada solicita un objeto de la fábrica abstracta, sin saber exactamente qué clase de objeto se devolverá. La implementación subyacente devuelta puede depender de una variedad de factores, como la configuración regional actual, el sistema operativo o configuración (Phillips, Giridhar, Kasampalis 2016). En la propuesta de solución se evidenció la utilización de este patrón mediante la clase Figure perteneciente a la clase *FigureCanvasQTAgg* cuya finalidad es la creación del objeto figura.

Singleton**:** Es un patrón estructural de diseño, planteado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella (Phillips, Giridhar, Kasampalis 2016). En la propuesta de solución se utilizó este patrón en la instancia de las clases, ejemplo de ello la instancia a las clases *Analog*, dando un punto de acceso global a la clase.

2.6 Diseño de Base Datos

El modelo entidad-relación es una técnica para definir las necesidades de información de su organización. Proporciona una buena base para sistemas de alta calidad dirigidos a satisfacer las necesidades de su empresa. De una forma más simple es la identificación de los asuntos de mayor importancia dentro de una organización (entidad), las propiedades de esos asuntos (atributos) y cómo se relacionan entre sí (relación) (Barker 1994).

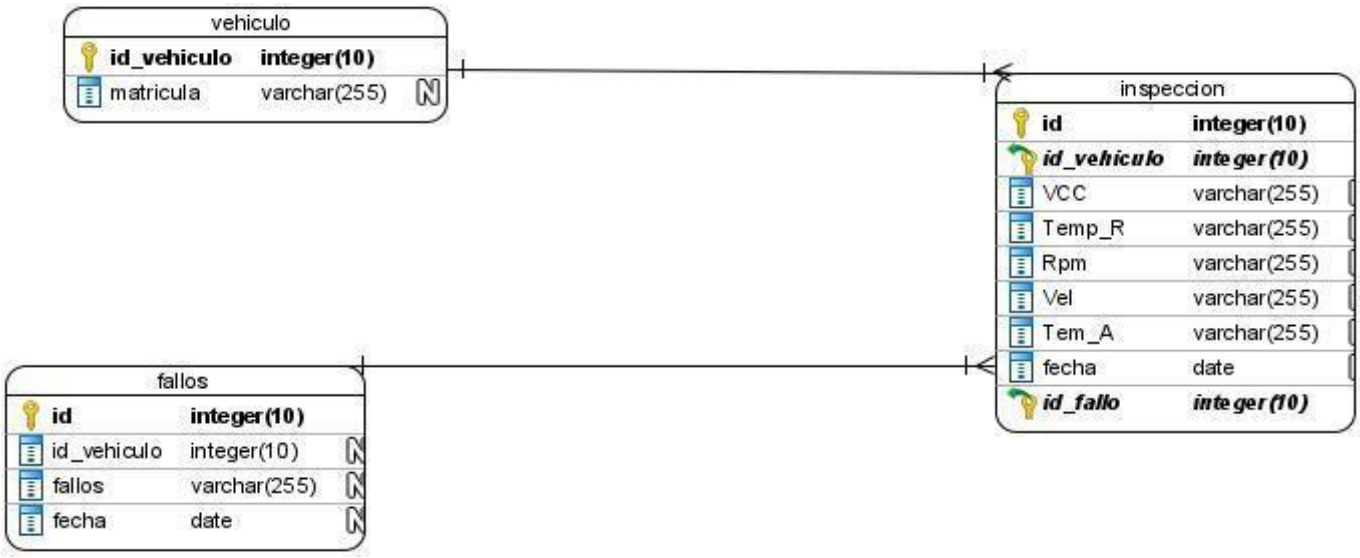


Figura 8 Diseño de base datos, Diagrama entidad-relación (Fuente: Elaboración propia).

Descripción de la Base de Datos

Tabla 12 Descripción de la base datos - tabla inspeccion

Nombre	Tipo	Descripción
id	int	Identificador único
id_vehiculo	int	Identificador del vehículo
VCC	varchar	Datos del valor de carga calculado
Temp_R	varchar	Datos de temperatura del refrigerante
Rpm	varchar	Datos de las revoluciones por minuto
Vel	varchar	Datos de la velocidad del vehículo

Tem_A	varchar	Datos de la temperatura ambiente
id_fallo	int	Identificador del arreglo de fallos
fecha	date	Fecha de la adquisición de los datos

Tabla 13 Descripción de la base datos- tabla fallos

Nombre	Tipo	Descripción
id	int	Identificador único
id_vehiculo	int	Identificador del vehículo
fallos	varchar	Arreglo de códigos de fallos
fecha	date	Fecha de la adquisición de los datos

Tabla 14 Descripción de la base datos- tabla vehículo

Nombre	Tipo	Descripción
id_vehiculo	id	Identificador único
matricula	varchar	Matrícula del vehículo

Conclusiones del capítulo

El análisis y diseño de la propuesta permitió realizar una descripción detallada de las características del sistema GesMot-OBD2, lo que permitió un mejor entendimiento para la fase de implementación al tener los principales artefactos para el desarrollo. Se definieron 10 requisitos funcionales y 22 no funcionales, los cuales proporcionan una guía de desarrollo de las funcionalidades del sistema. Se seleccionó la arquitectura MVC para el desarrollo de la solución porque esta responde al desarrollo del sistema. La definición del estilo arquitectónico y la realización, las historias de usuario y el diagrama de clase de diseño permitieron un mejor entendimiento del funcionamiento del sistema. Se seleccionaron los patrones GRASP y GoF a utilizar en la implementación para lograr una adecuada reutilización del código para futuras versiones.

CAPÍTULO 3: Pruebas y Validación del sistema informático para la adquisición de datos en vehículos mediante el dispositivo OBD2.

En este capítulo se documenta el proceso de implementación de los elementos identificados durante la elaboración del capítulo anterior. Para dar cumplimiento a este propósito, se modela el diagrama de componentes, diagrama de despliegue, las interfaces gráficas de usuario. Además, se incluyen los resultados de las pruebas y las validaciones realizadas a la aplicación desarrollada, lo cual permitió que los componentes desarrollados cumplan con los requisitos establecidos.

3.1 Implementación

El modelo de implementación describe cómo los elementos del diseño se implementan en componentes. También describe la organización de los componentes según los mecanismos de estructuración y modularización disponibles en el entorno de desarrollo, el lenguaje de programación utilizado, y la dependencia entre componentes. A partir de los resultados del análisis y el diseño se realiza la implementación del sistema, generando el modelo de implementación como artefacto de esta disciplina. Este modelo está conformado por el modelo de componentes y el modelo de despliegue.

Estándar de codificación

Para Python existen normas para la mejora de su codificación, los PEPs (*Python Enhancement Proposals*). De estos estándares, el PEP8, es “Guía de estilo para el código Python” (*PEP 8 – Style Guide for Python Code | peps.python.org*).

1. Sangría
   - ✓ Use 4 espacios por nivel de sangría.
2. Espacios
   - ✓ Los espacios son el método de sangría preferido.
   - ✓ Las pestañas deben usarse únicamente para mantener la coherencia con el código que está ya sangrado con tabulaciones.
   - ✓ Python no permite mezclar tabulaciones y espacios para la sangría.
3. Codificación del archivo origen
   - ✓ El código en la distribución central de Python siempre debe utilizar UTF-8 y no debe tener una declaración de codificación.
4. Importaciones
   - ✓ Las importaciones generalmente deben estar en líneas separadas.

- ✓ Las importaciones deben agruparse en el siguiente orden:
  - Importaciones de biblioteca estándar.
  - Importaciones de terceros relacionados.
  - Importaciones específicas de bibliotecas/aplicaciones locales.

3.2 Modelo de componentes

Los diagramas de componentes muestran las interacciones y relaciones de los componentes de un modelo. Entendiéndose como componente a una clase de uso específico, que puede ser implementada desde un entorno de desarrollo, ya sea de código fuente, binario o ejecutable, dichos componentes poseen tipo, y están unidos mediante relaciones de dependencia (Pressman 2021). A continuación, se presenta el diagrama de componentes del sistema.

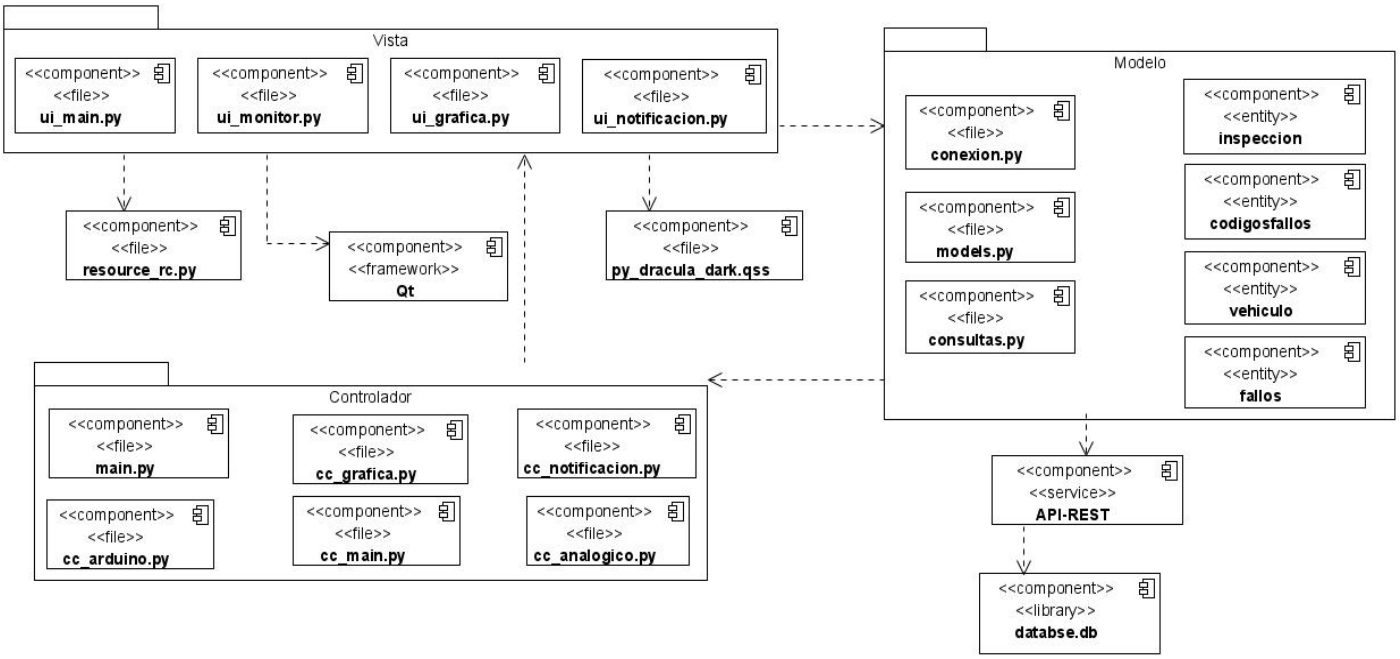


Figura 9 Diagrama de componentes del sistema (Fuente: Elaboración propia).

3.3 Diagrama de despliegue

El modelo de despliegue se realiza como parte de la implementación para describir la distribución física del sistema. Establece la correspondencia entre la arquitectura lógica, los procesos y los nodos. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar. Los nodos poseen relaciones que representan medios de comunicación entre

ellos (Pressman 2021). Se reflejan en este artefacto los protocolos de comunicación mediante los cuales se comunican los nodos respectivos.

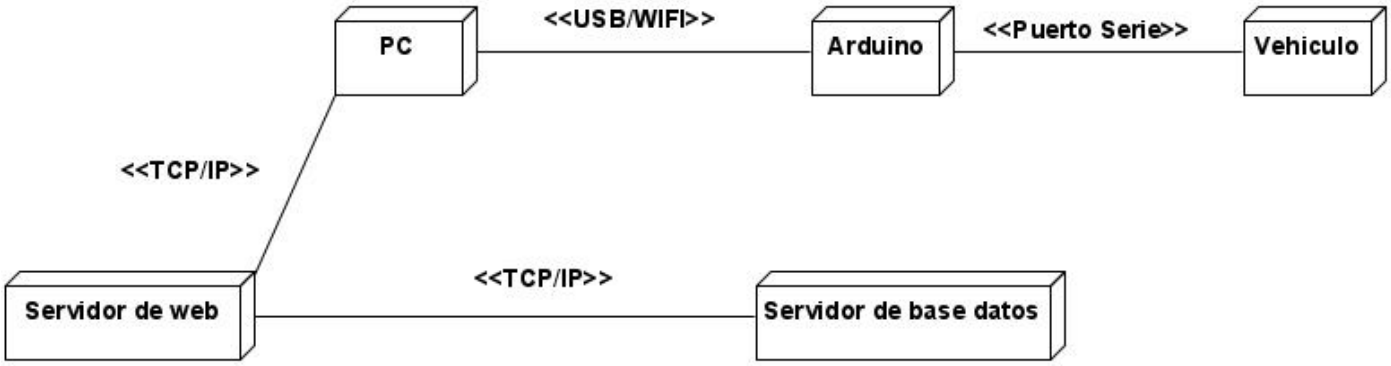


Figura 10 Diagrama de despliegue (Fuente: Elaboración propia).

3.4 Interfaz gráfica de usuario

A continuación, se muestra una representación de las interfaces gráficas que se tuvo como resultado en el proceso de desarrollo del software. Se muestra en la Figura 12 la interfaz de Inicio, la cual se muestra al iniciar el software.

Las demás interfaces gráficas se encuentran en el Anexo 4

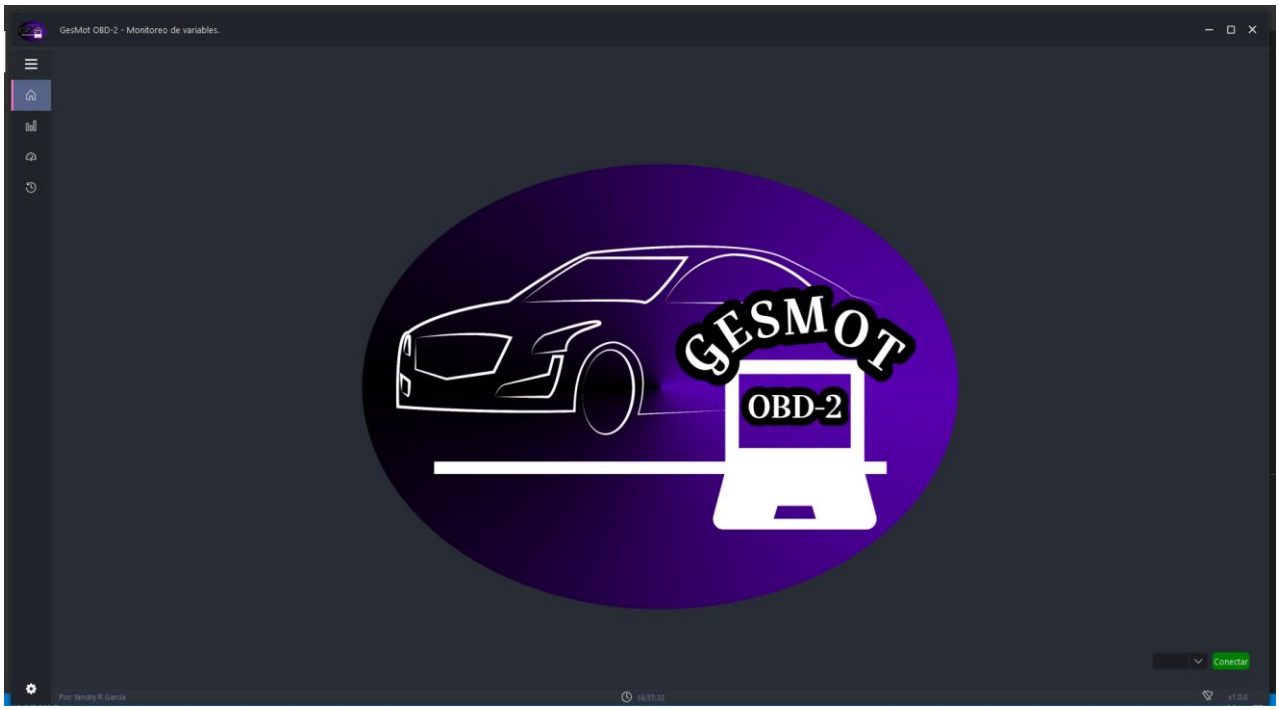


Figura 11 Interfaz gráfica Inicio (Fuente: Elaboración propia).

3.5 Pruebas de software

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia y el desempeño de un software. Las técnicas para encontrar problemas en un software son extensamente variadas y van desde el uso del ingenio por parte del personal ejecutor de las pruebas hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad (Valdivia Espinoza, Daniel Rolando, Valdivia Espinoza, Eduardo Geonias 2005). Se dividen en dos tipos de pruebas, Pruebas Funcionales y Pruebas no funcionales.

Sin duda alguna, la calidad es la piedra angular en todo, no sólo en el desarrollo de sistemas. No obstante, en esta ocasión revisaremos específicamente el desarrollo de software, y aquí es donde entra ISO/IEC 25000 (*SQuaRE — System and Software Quality Requirements and Evaluation*), la cual es una familia de normas que tiene por objetivo la creación de un marco de trabajo común para evaluar la calidad del producto software (*ISO 25010*).

Cuando hablamos de ISO 25010, estamos hablando de un modelo de calidad compuesto de 8 características que se relacionan con las propiedades estáticas del software y las propiedades

dinámicas del sistema informático. El modelo es aplicable tanto a los sistemas informáticos como a los productos de software.

Pruebas Funcionalidad

El sistema satisface las necesidades declaradas cuando se utilizan en condiciones específicas. Básicamente, es que el sistema haga lo que se desea que haga, o en otras palabras, es funcionalmente adecuado si cumple con todos los requisitos, los cubre correctamente y solo hace las cosas que son necesarias y adecuadas para completar las tareas (Elizabeth 2018).

Algunas pruebas para saber si cumplimos o no con la funcionalidad son las pruebas de regresión, pruebas del sistema, pruebas unitarias, pruebas de procedimientos, pruebas de aceptación, etc. Algunas de las aplicadas son:

- **Completitud funcional**

"Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados". Es decir, el sistema hace lo que se pide que haga, todas funciones que el cliente/usuario pide (ISO 25010).

C = rc/rs

rc = requisitos cumplidos

rs = requisitos solicitados

C = 10/10

C = 1

- **Pruebas de aceptación**

Las pruebas de aceptación son pruebas de caja negra definida por el cliente para cada historia de usuario, y tienen como objetivo asegurar que las funcionalidades del sistema cumplen con lo que se espera de ellas. Las pruebas de caja negra son un método de ensayo en el que los datos de prueba se derivan de los requisitos funcionales especificados sin tener en cuenta la estructura final del programa. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándose (Pressman 2021).

Las pruebas de aceptación son más importantes que las pruebas unitarias, dado que significan la satisfacción del cliente con el producto desarrollado y estas son creadas a partir de las Historias de Usuario.

- **Pruebas basadas en Historias de usuarios**

A continuación, aparece un ejemplo de las pruebas de aceptación realizadas a la solución, escogiendo la Historia de Usuario Gestionar conexión, el requisito Conectar dispositivo, las restantes se encuentran en la sección Anexo 5 Pruebas de aceptación.

Descripción General: *El Sistema debe Gestionar la conexión, poder conectar, seleccionar el puerto de conexión y detener la conexión.*

Condiciones de ejecución: *Debe estar un Arduino conectado por puerto USB*

Tabla 15 Pruebas de aceptación Historia de Usuario 1 (Fuente: Elaboración propia)

Escenario	Descripción	Puerto	Respuesta del sistema	Flujo central
El puerto es correcto	El sistema debe validar el puerto se es correcto	COM7	Se activa el icono de conexión	Se selecciona el puerto de conexión, el select y se da clic en el botón de conectar
El puerto es Incorrecto	El sistema no válida que el puerto es correcto	COM3	Se notifica que el puerto es incorrecto	
	El sistema válido que el puerto es vacío		Se notifica que el puerto está vacío	

- **Pruebas de regresión**

Las pruebas de regresión son pruebas de un programa previamente probado que ha sufrido modificaciones, para asegurarse que no se han introducido o descubierto defectos en áreas del software que no han sido modificadas como resultado de los cambios realizados. Se ejecuta cuando el software o su entorno han sido modificados. En el caso de la aplicación se realizaron las pruebas de regresión descritas al requisito funcional, salvar datos obtenidos y se arrojaron distintos resultados en dos distintas iteraciones

Tabla 16 Prueba de Regresión (Fuente: Elaboración Propia)

Iteración	Error a Resolver	Problemas Encontrados
Iteración 1	No era posible salvar datos	Al salvar datos no se obtenían el listado de datos requerido, se obtenían menos datos
Iteración 2	Se corrigió el error de la Iteración 1	No se encontraron problemas

- **Pruebas unitarias**

Es una forma de probar una unidad o fragmento de código más pequeño que se puede aislar lógicamente en un sistema (Miriam 2021).

Los principales puntos de estas pruebas son:

- Garantizar que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- Ejercitar todas las decisiones lógicas en las vertientes verdadera y falsa.
- Ejecutar todos los bucles en sus límites operacionales.
- Ejercitar las estructuras internas de datos para asegurar su validez.

Luego de realizar la prueba de caja blanca mediante la prueba del camino básico a la funcionalidad de guardar historial, se determinó que la complejidad aciclomatica era 4 por lo que el método es sencillo y de poco riesgo para el sistema (Pressman 2021).

v(G) = e – n + 2, donde se representa el número de aristas y n el número de nodos.

e = 11 n = 9

v(G) = número de regiones cerradas en el grafo + 1

v(G) = 11- 9 + 2

v(G) = 4

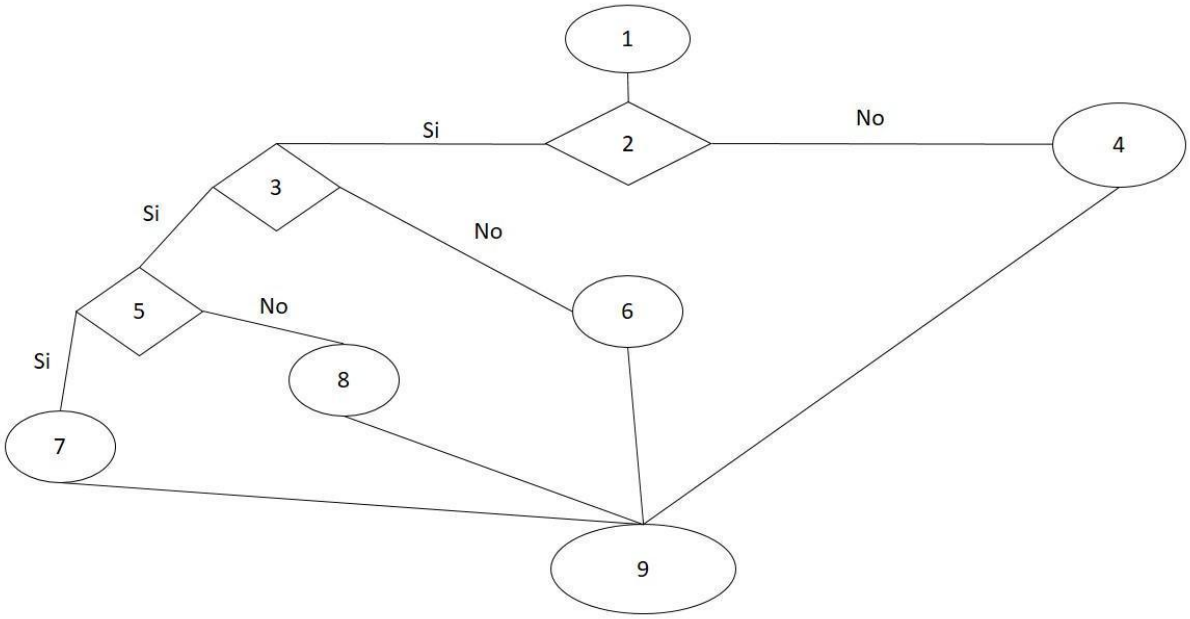


Figura 12 Prueba unitaria, Camino básico (Fuente: Elaboración propia)

Caminos:

1-2-3-5-7-9

1-2-4-9

1-2-3-6-9

1-2-3-5-8-9

En la siguiente figura se muestra un ejemplo de caso de prueba unitaria realizada al procedimiento obtener_vehiculos de la clase Vehiculo mediante la librería “pytest” de Python, en la cual en un

primer momento dio error, dado por la no conexión con la API-REST para obtener los datos. Dicho error fue solucionado en un segundo momento.

Tabla 17 Pruebas unitarias (Fuente: Elaboración propia)

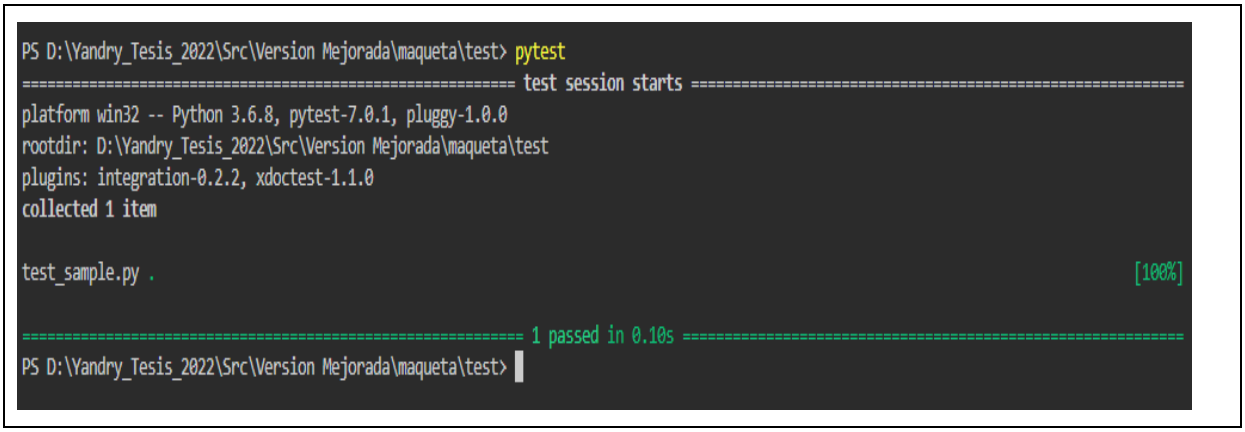
Caso de prueba unitaria
Iteración 1

```python
import requests

def test_obtener_vehiculo():
    path = "vehiculo"
    response = requests.get('http://127.0.0.1:8080/'+path)
    assert response.status_code == 200
```

```
PS D:\Yandry_Tesis_2022\Src\Version Mejorada\maqueta\test> pytest
============================= test session starts =============================
platform win32 -- Python 3.6.8, pytest-7.0.1, pluggy-1.0.0
rootdir: D:\Yandry_Tesis_2022\Src\Version Mejorada\maqueta\test
plugins: integration-0.2.2, xdoctest-1.1.0
collected 1 item

test_sample.py F                                                         [100%]

================================== FAILURES ===================================
_____________________________ test_obtener_vehiculo _____________________________

    def test_obtener_vehiculo():
        path = "vehiculo"
        response = requests.get('http://127.0.0.1:8080/'+path)
>       assert response.status_code == 200
E       assert 404 == 200
E        +  where 404 = <Response [404]>.status_code

test_sample.py:7: AssertionError
=========================== short test summary info ===========================
FAILED test_sample.py::test_obtener_vehiculo - assert 404 == 200
============================== 1 failed in 0.22s ==============================
PS D:\Yandry_Tesis_2022\Src\Version Mejorada\maqueta\test>
```

Iteración 2

```python
import requests

def test_obtener_vehiculo():
    path = "vehiculos"
    response = requests.get('http://127.0.0.1:8080/'+path)
    assert response.status_code == 200
```



Pruebas de Integración:

Nuestro software debe ser compatible con su entorno, por lo que este debe ser compatible con el hardware y software que lo rodea.

Por un lado, debe existir coexistencia, el cual se refiere a la capacidad de desempeñarse de manera eficiente al compartir un entorno y recursos comunes con otros, sin un impacto perjudicial en sí mismo o en otro. También debe existir interoperabilidad, lo que significa la posibilidad de intercambiar información y usar dicha información con otros sistemas. Debido a esto, es muy importante que desarrollemos basado en estándares, ya que estos mismos nos ayudan a sopesar estos problemas (*ISO 25010*).

Las pruebas que se pueden realizar para garantizar estos atributos de calidad son las pruebas de conformidad, compatibilidad, interoperabilidad y conversión. De estas se seleccionó ejecutar la prueba de interoperabilidad comprobando que el sistema interactúa a la par con otro sistema.

Para lograr la interoperabilidad del sistema se configuró el sistema para exportar los datos en formato JSON, el cual es un formato más factible para cuando se quiere transmitir datos a través de una red, permitiendo la conexión con otro sistema, de una manera más ágil.

Pruebas de rendimiento:

Evalúa el rendimiento, la velocidad o capacidad de respuesta de la aplicación que se prueba bajo la carga de trabajo requerida.

Sabemos que un sistema con alto rendimiento es rápido, escalable y estable incluso cuando hay una gran cantidad de usuarios concurrentes, por lo que debemos evaluar cómo se comporta el sistema en ciertas situaciones con diferentes tamaños de carga, es decir, que tenga un rendimiento de acuerdo a lo esperado, y con comportamiento, nos referimos a comportamiento en el tiempo de carga, uso de recursos, eficiencia y capacidad (Miriam 2021).

Algunas pruebas de rendimiento que podemos realizar, son de carga, resistencia, peak, etc.

Se aplicó una prueba de carga a los métodos para determinar el tiempo de duración de estos, a continuación, se muestra una tabla con el resultado de estos.

Tabla 18 Resultado de las pruebas de rendimiento, Pruebas de Carga

Métodos	Tiempo empleado
consultar_sensor_historial()	140 ms
iniciar_monitoreo()	1440 ms
guardar_historial()	800 ms

Tras analizar las métricas arrojadas en las pruebas de rendimiento, se puede concluir que el sistema responde en el tiempo solicitado en los requisitos de calidad según la norma ISO 25010. Con un tiempo inferior a los 3000 ms o 3 segundos.

Otra prueba realizada fue la prueba de resistencia, como se evidencia a continuación:

Tabla 19 *Resultados* de las pruebas de rendimiento, prueba de resistencia (Fuente: Elaboración Propia)

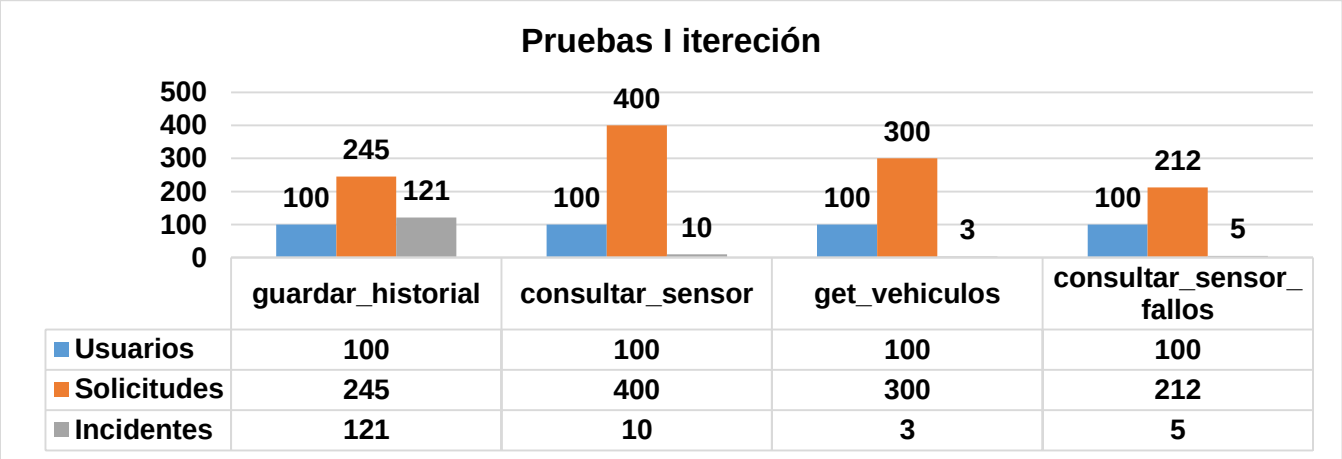
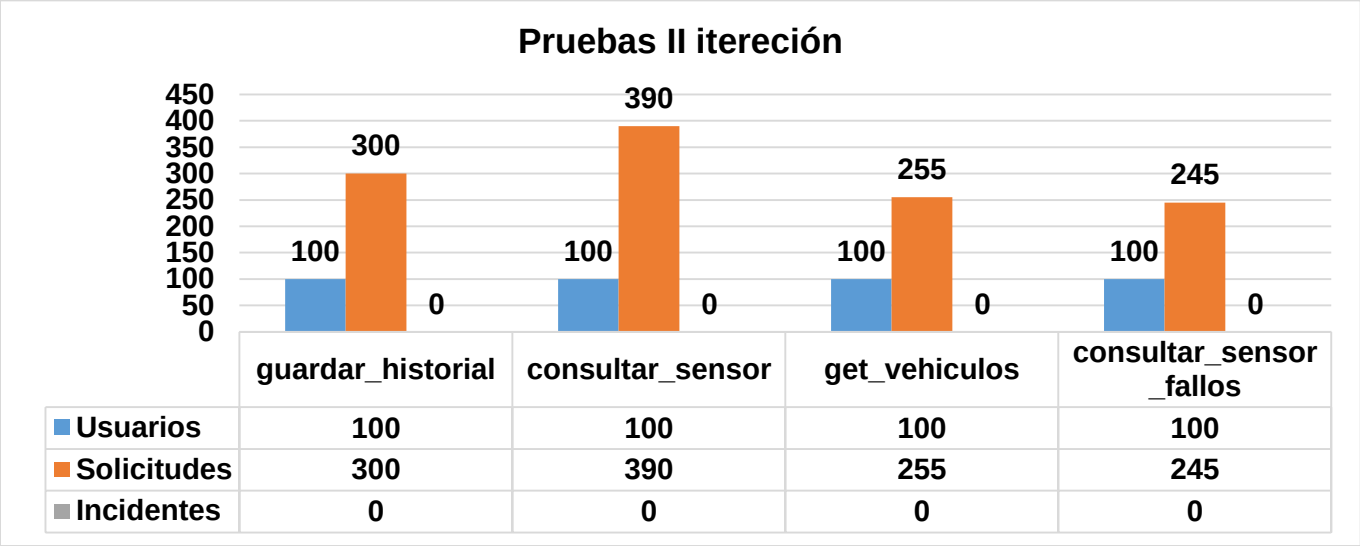


Tabla 20 Resultados de las pruebas de rendimiento, prueba de resistencia (Fuente: Elaboración Propia)



	guardar_historial	consultar_sensor	get_vehiculos	consultar_sensor_fallos
Usuarios	100	100	100	100
Solicitudes	300	390	255	245
Incidentes	0	0	0	0

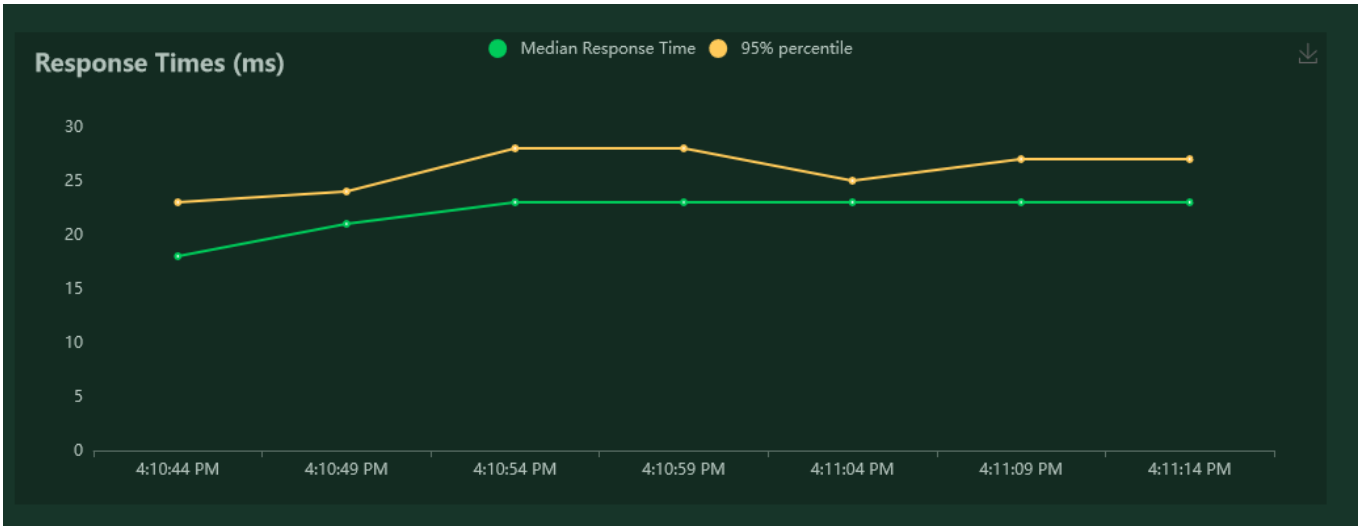


Figura 13 Prueba de resistencia Tiempo de respuesta por segundo, método guardar historial (Fuente: Test Report for Locus)

Resultado de la prueba de rendimiento

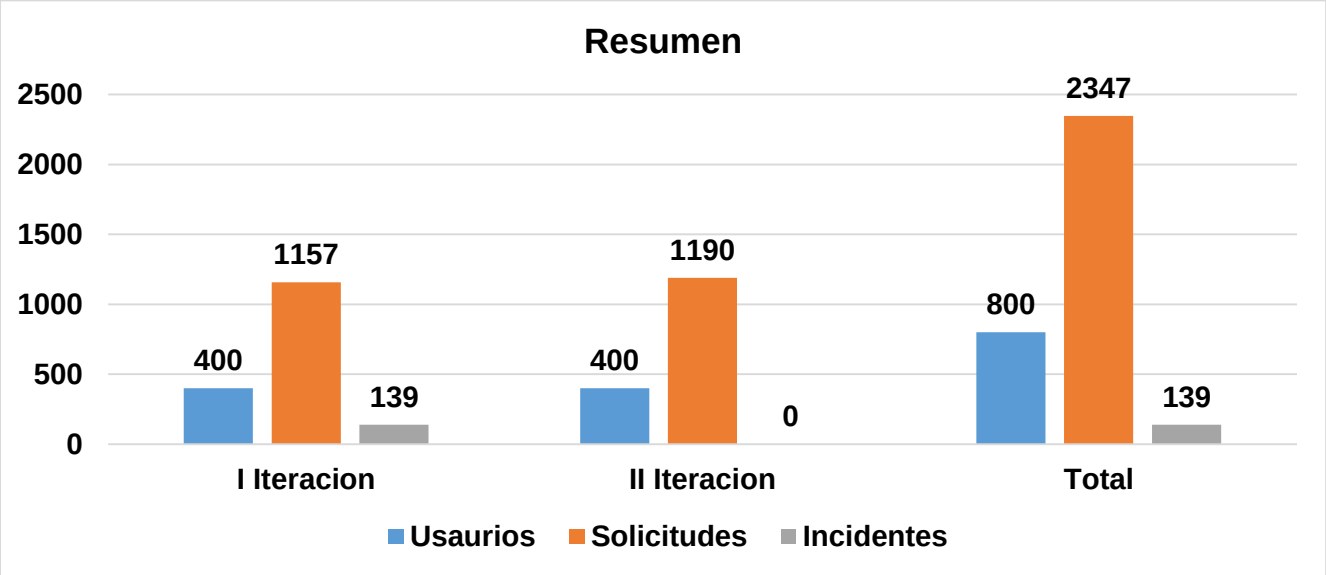


Figura 14 Resultado de las pruebas de rendimiento (Fuente: Elaboración propia)

Tras dos iteraciones, se encontraron en la primera iteración un conjunto de 139 incidentes, de los cuales se debieron al alto número de solicitudes por los usuarios y largo tiempo de respuesta que impedía responder varias peticiones a la vez, ya en la segunda se solucionaron estos problemas disminuyendo los tiempos de respuestas por petición.

Conclusiones del capítulo

En el capítulo concluido se elaboró el diagrama de componentes, lo que permitió visualizar con facilidad la estructura general de la solución.

Se establecieron los estándares de codificación a tener en cuenta para la implementación del GesMot-OBD2 lo que permitió garantizar que el código posea alta calidad, reducción de errores y pueda ser mantenido fácilmente. De igual forma, permite la reutilización por otros desarrolladores que lo necesiten. Se elaboró un diagrama de despliegue como guía para el correcto despliegue del sistema.

Se realizaron seis tipologías de pruebas a) Completitud Funcional, b) Aceptación, c) Regresión, d) Interoperabilidad, e) Carga de trabajo y f) Resistencia. De forma general se detectaron un nivel de no conformidades, todas resueltas en las diferentes iteraciones ejecutadas.

CONCLUSIONES FINALES

Considerando los resultados descritos en este informe, la necesidad y el objetivo planteado por la investigación se arriban a las siguientes conclusiones:

- ✓ Mediante el método histórico - lógico y analítico - sintético se evaluó el criterio de diferentes autores en diversas fuentes bibliográficas, sobre los procesos de diagnóstico de vehículos enfatizando el marco teórico sobre los principales problemas y soluciones existentes en cuanto a la adquisición de datos, logrando identificar las características que describen los sistemas y su adaptación al contexto nacional.
- ✓ A través del método empírico se elaboró una entrevista y una encuesta logrando identificar funcionalidades y características necesarias para el diseño e implementación de una aplicación informática de adquisición de datos mediante el dispositivo ODB2.
- ✓ Las técnicas de modelado posibilitaron el diseño de la aplicación, mediante los artefactos ingenieriles, que propone la metodología AUP-UCI en su escenario 4, favoreciendo la planificación de todo el proyecto, ahorrando tiempo y recursos.
- ✓ Se desarrolló una aplicación para la adquisición de datos en vehículos mediante una placa Arduino UNO como simulador de dispositivo OBD2, que permite reducir los gastos por mantenimiento y tener un mayor control sobre la emisión de CO_2 en los vehículos.
- ✓ La realización de pruebas de software permitió verificar y validar el correcto funcionamiento de la aplicación. Estas pruebas arrojaron resultados satisfactorios en relación con el código y el conjunto de interfaces implementadas.

RECOMENDACIONES

Antes de finalizar, deseamos sugerir algunas recomendaciones en base a los resultados y las conclusiones a que se llegó luego del presente estudio:

- ✓ Adaptar a otras plataformas (Android).
- ✓ Realizar un sistema para la evaluación de los datos adquiridos, que contribuya a la toma de decisiones.
- ✓ Por último, se recomienda extender el uso del sistema en el país, por su alta rentabilidad ante la toma de decisiones con vista al mantenimiento de los vehículos.

Referencias Bibliográficas

ACITI, Claudio, URRACO, Mauricio y TODOROVICH, Elías, 2017. Prototipo de sistema de captura y monitoreo de datos OBD-II de vehículos. En: *XXIII Congreso Argentino de Ciencias de la Computación (La Plata, 2017).* en línea. octubre 2017. [Accedido 28 junio 2022]. ISBN 978-950-34-1539-9. Recuperado a partir de: http://sedici.unlp.edu.ar/handle/10915/63713

AGUERO, Mónica B, 2017. Adquisicion-de-datos. en línea. 2017. [Accedido 7 noviembre 2022]. Recuperado a partir de: http://materias.df.uba.ar/mta2019c1/files/2014/08/Adquisicion-de-datos-Aguero.pdf

ANASTASIO, Meseguer y ENRIQUE, Javier, 2013. Caracterización de los estilos de conducción mediante smartphones, dispositivos obd-ii y redes neuronales. en línea. 12 febrero 2013. [Accedido 4 mayo 2022]. Recuperado a partir de: https://riunet.upv.es/handle/10251/21025Accepted: 2013-02-12T07:41:31Z

Arduino | Arduino.cl, 2019. en línea. [Accedido 7 noviembre 2022]. Recuperado a partir de: https://arduino.cl/

BAHIT, Eugenia, 2020. *Python para Principiantes: Edición 2020*. EBRC Publisher. ISBN 978-1-83819-011-8.

BARKER, Richard, 1994. *El modelo entidad-relación CASE*methodtm*. Ediciones Díaz de Santos. ISBN 978-0-201-60111-4. Google-Books-ID: hbOTo05ddxAC

BECK, Kent, 2000. *extreme programming eXplained: embrace change*. Reading, MA: Addison-Wesley. ISBN 978-0-201-61641-5. QA76.76.D47 B434 2000

BLOG, Requeridos, 2018. Requerimientos Funcionales y No Funcionales, ejemplos y tips. *Medium*. en línea. 29 noviembre 2018. [Accedido 13 septiembre 2022]. Recuperado a partir de: https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

Codifica en Python version 14-08-2020 WEB.pdf, sin fecha. en línea. [Accedido 13 septiembre 2022]. Recuperado a partir de:

https://dspace.ups.edu.ec/bitstream/123456789/19346/4/Codifica%20en%20Python%20version%2014-08-2020%20WEB.pdf

CONTRERA RAMÍREZ, José Antonio, 2020. *Usos del Puerto OBD2 para diagnóstico del motor de un vehículo desde un dispositivo móvil*. en línea. Tecnologo Nacional de Mexico: Instituto Tecnológico de Huejutla. Recuperado a partir de: https://rinacional.tecnm.mx/bitstream/TecNM/1127/1/JOSE%20ANTONIO%20CONTRERAS%20RAMIREZ.pdf

CÓRDOVA, Tenorio y MAGALLANES, 2020. *"Análisis y diseño de un sistema de seguridad y monitoreo vehicular usando dispositivos OBD2 GSM"*. en línea. Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniería en Networking y Telecomunicaciones. [Accedido 31 marzo 2022]. Recuperado a partir de: http://repositorio.ug.edu.ec/handle/redug/49483Accepted: 2020-11-04T15:38:52Z

CRESPO, Enrique, 2016. DAQ | Aprendiendo Arduino. en línea. 2016. [Accedido 7 noviembre 2022]. Recuperado a partir de: https://aprendiendoarduino.wordpress.com/tag/daq/

Cuba - Emisiones de CO_2 del consumo de combustibles sólidos (kilotoneladas), sin fecha. en línea. [Accedido 21 noviembre 2022]. Recuperado a partir de: https://www.indexmundi.com/es/datos/cuba/indicador/EN.ATM.CO_2E.SF.KT

DART, S. A., ELLISON, R. J., FEILER, P. H. y HABERMANN, A. N., 1987. Software Development Environments. *Computer*. en línea. 1 noviembre 1987. Vol. 20, no. 11, pp. 18-28. [Accedido 7 junio 2022]. DOI 10.1109/MC.1987.1663413.

Definición de RPM - Qué es, Significado y Concepto, sin fecha. en línea. [Accedido 4 mayo 2022]. Recuperado a partir de: https://definicion.de/rpm/

DIMATÉ CÁCERES, Juan Manuel y GONZÁLEZ CASTILLO, Pedro Mauricio, 2013. Diseño de una interfaz gráfica en Labview para el diagnostico de vehículos por medio de OBD2. *instname:Universidad Pontificia Bolivariana*. en línea. 28 agosto 2013. [Accedido 4 mayo 2022]. Recuperado a partir de: https://repository.upb.edu.co/handle/20.500.11912/911Accepted: 2013-08-28T16:59:43Z

ELIZABETH, Carrión Vaca Gabriela, 2018. COMPARATIVA DE TRES HERRAMIENTAS DE REALIDAD AUMENTADA UTILIZANDO UNA METODOLOGÍA DE MEDICIÓN DE SOFTWARE ISO 25010. . 2018. pp. 184.

Emisiones de CO_2 originadas por el transporte (% del total de la quema de combustible) - Cuba | Data, sin fecha. en línea. [Accedido 15 noviembre 2022]. Recuperado a partir de: https://datos.bancomundial.org/indicador/EN.CO_2.TRAN.ZS?end=2014&locations=CU&start=2014&type=shaded&view=map

Emissions Gap Report 2020, 2021. . United Nations. ISBN 978-92-807-3812-4.

FERNANDEZ, Arturo, 2013. *Python 3 al descubierto - 2a ed.* Alfaomega Grupo Editor. ISBN 978-607-622-008-5. Google-Books-ID: f4BNDAAAQBAJ

GARCIA ALCIVAR, Bolivar Xavier, 2015. *Analisis y deseño de un sistema para el proceso de esconaeo de los vehiculos*. en línea. Guayaquil. [Accedido 3 mayo 2022]. Recuperado a partir de: https://dspace.ups.edu.ec/bitstream/123456789/10311/1/UPS-GT001210.pdf

GARCÍA FRANCO, Vilma, GARCÍA NÚÑEZ, Rubén Darío, LORENZO GONZÁLEZ, Marisela, HERNÁNDEZ CABEZAS, Marilys, GARCÍA FRANCO, Vilma, GARCÍA NÚÑEZ, Rubén Darío, LORENZO GONZÁLEZ, Marisela y HERNÁNDEZ CABEZAS, Marilys, 2020. Los mapas conceptuales como instrumentos útiles en el proceso enseñanza-aprendizaje. *MediSur*. en línea. diciembre 2020. Vol. 18, no. 6, pp. 1154-1162. [Accedido 7 noviembre 2022]. Recuperado a partir de: http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S1727-897X2020000601154&lng=es&nrm=iso&tlng=es

GARCÍA MONTERO, Miguel, 2013. *FACULTAD DE MECÁNICA ESCUELA DE INGENIERÍA AUTOMOTRIZ*. en línea. Recuperado a partir de: dspace.espoch.edu.ec/bitstream/123456789/2609/1/65T00068.pdf

GARCÍA PEÑALVO, Francisco José y GARCÍA HOLGADO, Alicia, 2017. Modelo de dominio. . 2018 2017. pp. 35.

GODOY, Yadelis Velázquez, CINTRA, Alionuska Velázquez y ROLO, Lester Collado, sin fecha. Diseño de herramienta para casos de pruebas funcionales en la Universidad de las Ciencias Informáticas. . pp. 16.

HMONG.WIKI, sin fecha. PID OBD-II ModosyPID estándar. en línea. [Accedido 7 julio 2022]. Recuperado a partir de: https://hmong.es/wiki/OBD-II_PIDs

Introduction to PySide toolkit, sin fecha. en línea. [Accedido 8 septiembre 2022]. Recuperado a partir de: https://zetcode.com/gui/pysidetutorial/introduction/

ISO 25010, sin fecha. en línea. [Accedido 8 septiembre 2022 a]. Recuperado a partir de: https://iso25000.com/index.php/normas-iso-25000/iso-25010

ISO 25010, sin fecha. en línea. [Accedido 8 septiembre 2022 b]. Recuperado a partir de: https://iso25000.com/index.php/normas-iso-25000/iso-25010?start=3

IVÁN CISNEROS RODRÍGUEZ, sin fecha. *Los sensores en el automóvil*. en línea. [Accedido 9 junio 2022]. Recuperado a partir de: http://www.tutallermecanico.com.mx/Uploads/TP11-03.pdf

JACOBSON, Ivar, BOOCH, Grady y RUMBAUGH, James, 2006. *El proceso unificado de desarrollo de software*. Madrid: Addison Wesley. ISBN 978-84-7829-036-9.

Journal of Innovation in Computer Science and Engineering, sin fecha. . pp. 11.

KHAN, Inayat, KHUSRO, Shah y ALAM, Iftikhar, 2019. Smartphone Distractions and its Effect on Driving Performance using Vehicular Lifelog Dataset. En: *2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. en línea. Swat, Pakistan: IEEE. julio 2019. pp. 1-6. [Accedido 1 abril 2022]. ISBN 978-1-72813-825-1. DOI 10.1109/ICECCE47252.2019.8940697.

KOWALIK, Bartosz, 2018. Introduction to car failure detection system based on diagnostic interface. En: *2018 International Interdisciplinary PhD Workshop (IIPhDW)*. en línea. Swinoujście: IEEE. mayo 2018. pp. 4-7. [Accedido 1 abril 2022]. ISBN 978-1-5386-6143-7. DOI 10.1109/IIPHDW.2018.8388233.

LA IMPORTANCIA DE LAS PRUEBAS UNITARIAS PARA COMPROBAR FRAGMENTOS DE CÓDIGO - Cero Ideas, sin fecha. en línea. [Accedido 21 septiembre 2022]. Recuperado a partir de: https://ceroideas.es/la-importancia-de-las-pruebas-unitarias-para-comprobar-fragmentos-de-codigo/

MALOY SMITH, Grant, 2021. ¿Qué es el bus CAN (red de área del controlador) y cómo se compara con otras redes de bus de vehículos? *dewesoft.com*. en línea. 6 mayo 2021. Recuperado a partir de: https://dewesoft.com/es/daq/que-es-el-bus-can#:~:text=La%20red%20de%20%C3%A1rea%20del,confiable%20y%20basada%20en%20prioridades.

MANAMPERI, Vidushan, LIYAARACHCHI, Kavindu, SAMARANAYAKE, Sadeepa, FERNANDO, Vimukthi, ABEYGUNAWARDHANA, Pradeep y DE SILVA, Rajitha, 2021. VEHlA: Safety enhancement of non-premium vehicle assisting system for personal vehicle users. En: *2021 6th International Conference on Inventive Computation Technologies (ICICT)*. en línea. Coimbatore, India: IEEE. 20 enero 2021. pp. 892-899. [Accedido 1 abril 2022]. ISBN 978-1-72818-501-9. DOI 10.1109/ICICT50816.2021.9358697.

MIRIAM, 2021. Qué es el testing de software. *Profile Software Services*. en línea. 7 septiembre 2021. [Accedido 21 septiembre 2022]. Recuperado a partir de: https://profile.es/blog/que-es-el-testing-de-software/

Modelo entidad relación: descripción y aplicaciones, sin fecha. en línea. [Accedido 8 septiembre 2022]. Recuperado a partir de: https://www.esic.edu/rethink/tecnologia/modelo-entidad-relacion-descripcion-aplicaciones

MONTESDEOCA VIVANCO, Andrea Gabriela, 2021. Implementación de un prototipo electrónico para registros telemáticos y detección de fallos en motores de automóviles mediante sistema OBD II. *30 de Juni 2021*. en línea. julio 2021. Vol. 1, pp. 68-87. Recuperado a partir de: https://tesla.puertomaderoeditorial.com.ar/index.php/tesla/article/download/6/8

MORENO, Miguel Ortega y GÓMEZ, Genoveva López, sin fecha. SISTEMA DE EVALUACIÓN DE LA CALIDAD DE LOS COMPONENTES WEB CENTRADO EN LOS USUARIOS FINALES. . pp. 114.

ORELLANA-ROMERO, Eric y DURAN-FAUNDEZ, Cristian, sin fecha. Introducci´on al desarrollo de GUI’s mediante Qt. . pp. 13.

PEP 8 – Style Guide for Python Code | peps.python.org, sin fecha. en línea. [Accedido 13 septiembre 2022]. Recuperado a partir de: https://peps.python.org/pep-0008/

PÉREZ GUERRA, Mayeleiny y RONDÓN MILANÉS, Raúl Alejandro, 2012. Extensión de la herramienta Visual Paradigm for UML para la administración de requisitos. en línea. 2012. [Accedido 7 junio 2022]. Recuperado a partir de: https://repositorio.uci.cu/jspui/handle/ident/TD_05808_12Accepted: 2016-09-14T19:17:21Z

PRESSMAN, Roger S, 2021. INGENIERÍA DE SOFTWARE. 9ª EDICION. *casadellibro*. en línea. 27 abril 2021. [Accedido 7 noviembre 2022]. Recuperado a partir de: https://www.casadellibro.com/libro-ingenieria-de-software-9-edicion/9781456287726/12328822

Protocolos de conexión OBD2, sin fecha. en línea. [Accedido 4 mayo 2022]. Recuperado a partir de: https://obd2-elm327.es/protocolos-conexion-obd2

Pyqt vs PySide (Spanish), sin fecha. *DEV Community*. en línea. [Accedido 8 septiembre 2022]. Recuperado a partir de: https://dev.to/amigosmaker/pyqt-vs-pyside-spanish-2kmg

Python: qué es y por qué deberías aprender a utilizarlo, sin fecha. en línea. [Accedido 5 mayo 2022]. Recuperado a partir de: https://www.becas-santander.com/es/blog/python-que-es.html

PYTHON, Real, sin fecha. Qt Designer and Python: Build Your GUI Applications Faster – Real Python. en línea. [Accedido 5 mayo 2022]. Recuperado a partir de: https://realpython.com/qt-designer-python/

Qué es MVC, sin fecha. en línea. [Accedido 7 noviembre 2022]. Recuperado a partir de: https://desarrolloweb.com/articulos/que-es-mvc.html

¿Qué es un requerimiento funcional? - La Oficina de Proyectos de Informática, sin fecha. en línea. [Accedido 13 septiembre 2022]. Recuperado a partir de: http://www.pmoinformatica.com/2018/05/que-es-requerimiento-funcional.html

RAMOS CORIA, Daniel Andrés, 2014b. Diseño de un Sistema de Monitoreo OBD-II con comunicación GSM. en línea. 29 abril 2014. [Accedido 2 abril 2022]. Recuperado a partir de: http://www.ptolomeo.unam.mx:8080/xmlui/handle/132.248.52.100/3407Accepted: 2014-04-29T16:17:46Z

RAMOS CORIA, Daniel Andrés, 2014a. Diseño de un Sistema de Monitoreo OBD-II con comunicación GSM. en línea. 29 abril 2014. [Accedido 9 junio 2022]. Recuperado a partir de:

http://www.ptolomeo.unam.mx:8080/xmlui/handle/132.248.52.100/3407Accepted: 2014-04-29T16:17:46Z

Reclu IT, sin fecha. en línea. [Accedido 5 mayo 2022]. Recuperado a partir de: https://recluit.com/que-es-visual-studio-code/

Repositório Institucional da Universidade Tecnológica Federal do Paraná (RIUT): Leitura OBD2 através de smartphone, sin fecha. en línea. [Accedido 1 abril 2022]. Recuperado a partir de: http://riut.utfpr.edu.br/jspui/handle/1/19812

RODRÍGUEZ MUNIVE, Mateo Esteban, 2021. Desarrollo de ciclos de conducción en la ciudad de Quito-Ecuador para un vehículo categoría M1. en línea. agosto 2021. [Accedido 1 abril 2022]. Recuperado a partir de: http://localhost:8080/xmlui/handle/123456789/4249Accepted: 2021-08-18T19:00:01Z

ROJAS, Nicolás F. Ormeño, 2019. ISO 25010 y el desarrollo de software. *Medium*. en línea. 15 mayo 2019. [Accedido 24 octubre 2022]. Recuperado a partir de: https://normeno.medium.com/iso-25010-y-el-desarrollo-de-software-112393a4b341

SCHOOL, Euroinnova Business, 2022. ¿Qué son las aplicaciones de escritorio? | Euroinnova. *Euroinnova Business School*. en línea. 2022. [Accedido 7 noviembre 2022]. Recuperado a partir de: https://www.euroinnova.edu.es/blog/aplicaciones-de-escritorio

SHAFI, Uferah, SAFI, Asad, SHAHID, Ahmad Raza, ZIAUDDIN, Sheikh y SALEEM, Muhammad Qaiser, 2018. Vehicle Remote Health Monitoring and Prognostic Maintenance System. *Journal of Advanced Transportation*. en línea. 2018. Vol. 2018, pp. 1-10. [Accedido 1 abril 2022]. DOI 10.1155/2018/8061514.

SIMBAÑA COYAGO, Wilson Andrés, 2015. Diseño e implementación de una solución telemática basada en OBD-II (On-Board Diagnostic) que permita obtener y procesar la información de los sensores del motor de un automóvil. en línea. 25 mayo 2015. [Accedido 3 mayo 2022]. Recuperado a partir de: http://bibdigital.epn.edu.ec/handle/15000/10659Accepted: 2015-06-04T21:12:57Z

SIMBAÑA, Wilson, CAIZA, Julio, CHÁVEZ, Danilo y LOPEZ, Gabriel, 2016. Diseño e Implementación de un Sistema de Monitoreo Remoto del Motor de un Vehículo basado en Obd-II y la plataforma Arduino. *Revista Politécnica*. en línea. 15 marzo 2016. Vol. 37, no. 1, pp. 25-25.

[Accedido 4 mayo 2022]. Recuperado a partir de: https://revistapolitecnica.epn.edu.ec/ojs2/index.php/revista_politecnica2/article/view/573

Sistema informático, sin fecha. *Economipedia*. en línea. [Accedido 30 mayo 2022]. Recuperado a partir de: https://economipedia.com/definiciones/sistema-informatico.html

SOCASI, Alonso Rodrigo Anchapaxi, 2016. *RECOLECCIÓN DE DATOS DEL SISTEMA OBD II DE UN AUTOMÓVIL USANDO UN DISPOSITIVO ANDROID*. en línea. Quito. Recuperado a partir de: https://bibdigital.epn.edu.ec/bitstream/15000/16553/1/CD-7218.pdf

SOLÍS, SAÚL OSBORN ESPINOSA y ISRAEL, CERVANTES, 2010. *ESCÁNER AUTOMOTRÍZ DE PANTALLA TÁCTIL T E S I S QUE PARA OBTENER EL TÍTULO DE: INGENIERO EN COMUNICACIONES Y ELECTRONICA - PDF Free Download*. en línea. [Accedido 4 mayo 2022]. Recuperado a partir de: https://docplayer.es/1287330-Escaner-automotriz-de-pantalla-tactil-t-e-s-i-s-que-para-obtener-el-titulo-de-ingeniero-en-comunicaciones-y-electronica.html

SQLite Home Page, sin fecha. en línea. [Accedido 9 junio 2022]. Recuperado a partir de: https://www.sqlite.org./index.html

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA, CLUJ-NAPOCA, 400114, ROMANIA, MUREŞAN, Vlad, ABRUDEAN, Mihail, SITA, Valentin, CLITAN, Iulia, UNGUREŞAN, Mihaela-Ligia y CARMEN CORDOŞ, Roxana, 2019. Modeling the Speed Dynamics of a Car. *International Journal of Modeling and Optimization*. en línea. febrero 2019. Vol. 9, no. 1, pp. 24-29. [Accedido 1 abril 2022]. DOI 10.7763/IJMO.2019.V9.678.

VALDIVIA ESPINOZA, DANIEL ROLANDO y VALDIVIA ESPINOZA, EDUARDO GEONIAS, 2005. *Estándares de calidad para pruebas de software*. en línea. Lima-Peru: Universidad Nacional Mayor de San Marco. [Accedido 21 septiembre 2022]. Recuperado a partir de: https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/272/Valdivia_ee.pdf?sequence=1&isAllowed=y

VILLATORO, Francisco R., 2019. Las emisiones globales de CO_2 fósil continúan desaforadas. *La Ciencia de la Mula Francis*. en línea. 6 diciembre 2019. [Accedido 21 noviembre 2022]. Recuperado a partir de: https://francis.naukas.com/2019/12/06/las-emisiones-globales-de-CO_2-fosil-continuan-desaforadas/

Visual Paradigm Product Overview, sin fecha. en línea. [Accedido 5 mayo 2022]. Recuperado a partir de: https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html

Visual Studio Code: Funcionalidades y extensiones, 2018. *El Blog de Aitana – Partner Microsoft y Sage en España*. en línea. [Accedido 7 junio 2022]. Recuperado a partir de: https://blog.aitana.es/2018/10/16/visual-studio-code/

What is PyQt? | Learn Python PyQt, sin fecha. en línea. [Accedido 5 mayo 2022]. Recuperado a partir de: https://pythonpyqt.com/what-is-pyqt/

ANEXOS

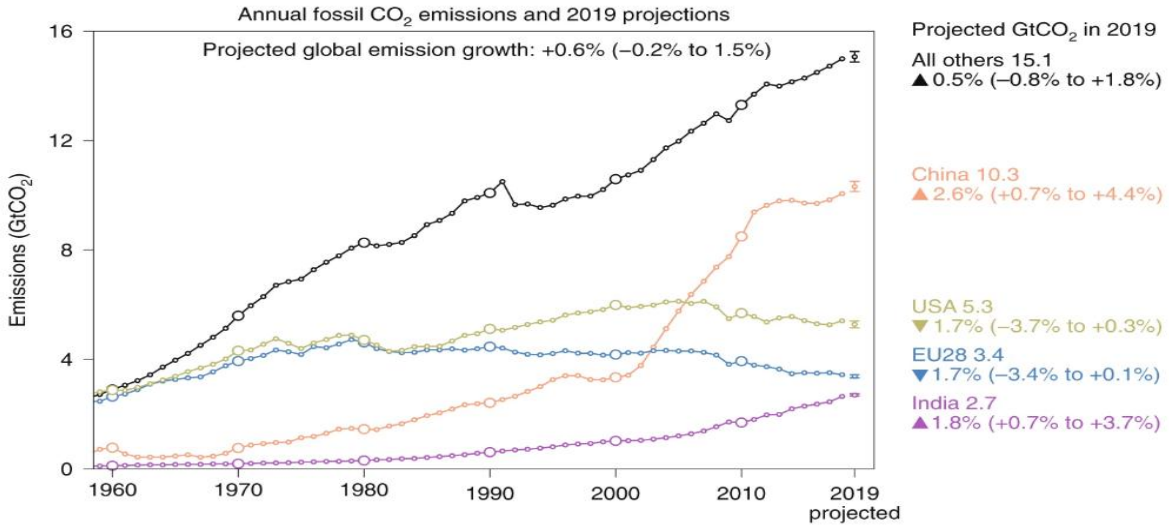


Figura 15 Grafico de emisiones de Gases de efecto invernadero desde 1960-2019 [Tomado de:(Villatoro 2019)]

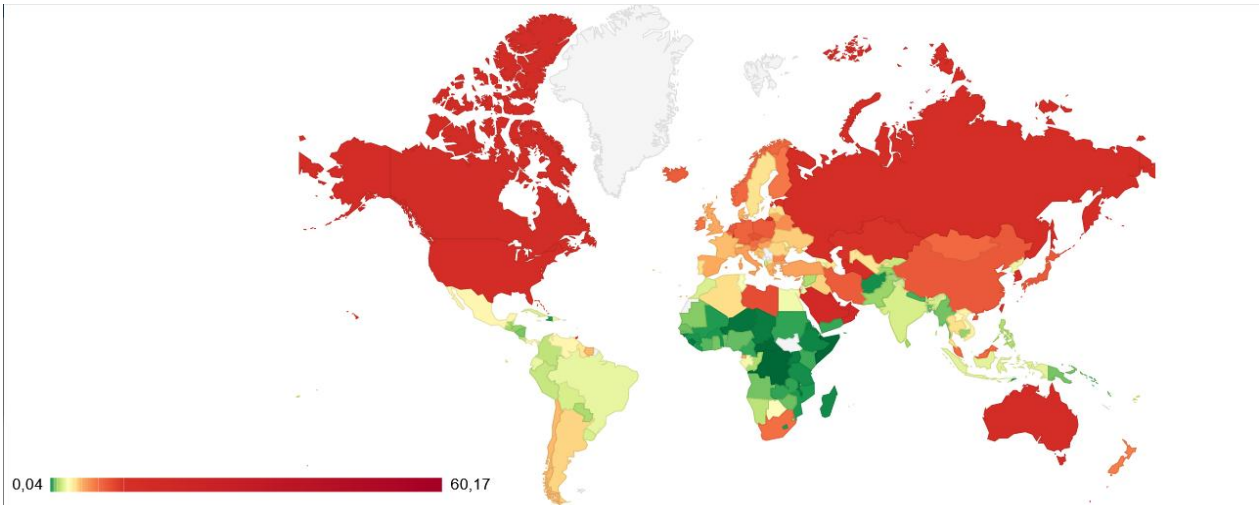


Figura 16 Mapa por países de emisión de CO_2(%) en trasporte en 2019 [Tomado de:(https:// datosmacro.com)

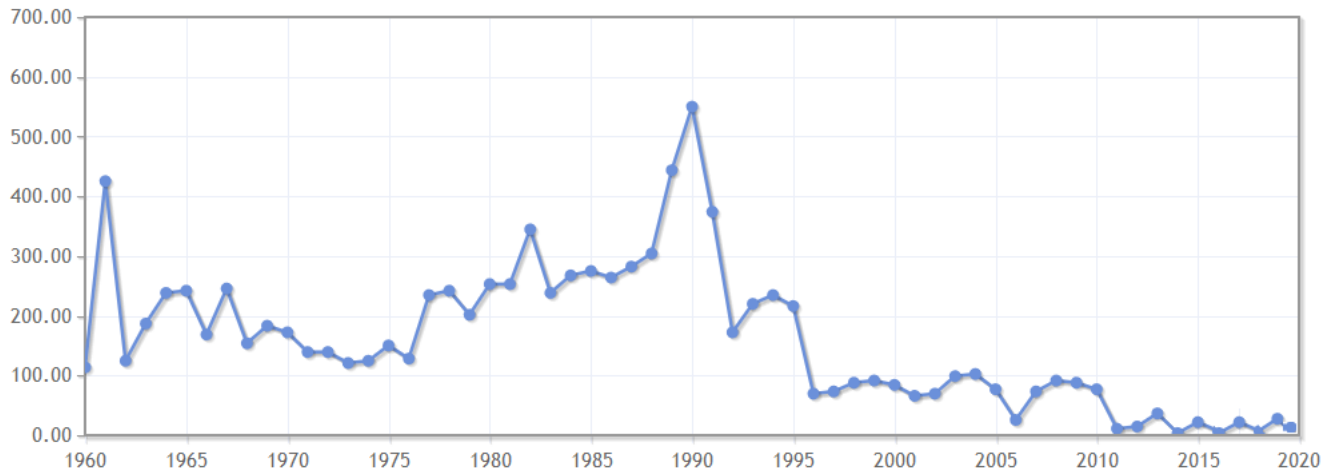


Figura 17 Emisiones de CO_2 en Cuba 1960-2020 (kilotoneladas) [Tomado de : https://www.indexmundi.com/es/datos/cuba/indicador/EN.ATM.CO_2E.SF.KT)]

Anexo 1 Obtención de Requisitos

Entrevista:

Objetivo: obtener información sobre los principales datos que son necesarios incorporar en la propuesta de solución.

- ✓ ¿Qué datos considera usted imprescindibles en la propuesta de solución?
- ✓ ¿Qué método de conexión considera importante para la propuesta de conexión?
- ✓ ¿Cree necesario solo mostrar datos de historial?
- ✓ ¿Cuáles son funciones que considera necesaria para su sistema?
- ✓ ¿Cuáles son los aspectos más relevantes que considera para su sistema?

Guía de estudio de la documentación:

Objetivo: comprender cómo se realiza el proceso de adquisición de datos, los datos adquiridos.

- ✓ ¿Qué datos son recolectados durante el proceso de adquisición?
- ✓ ¿Hay algún dato no obligatorio?
- ✓ ¿Cómo se realiza la adquisición?

Anexo 2 Historias de usuario.

Tabla 21 HU Gestionar adquisición de datos (Fuente: Elaboración propia).

Historia de usuario	
Número: 2	**Número del requisito:** Gestionar adquisición de datos

Programador: Yandry R García Pérez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 96 h
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 130 h

Descripción:

- **Objetivo:** Adquirir los datos de los sensores del vehículo.
- **Precondiciones:** Se debe tener conectado el dispositivo al computador.
- **Acciones a realizar:**

Iniciar adquisición:

Al dar clic en el **botón iniciar monitoreo** se iniciará la adquisición de datos de los sensores y códigos de fallos.

Finalizar adquisición:

Al dar clic en el **botón detener monitoreo** se detendrá la adquisición de datos de los sensores y códigos de fallos.

Observación:

Prototipo de interfaz:

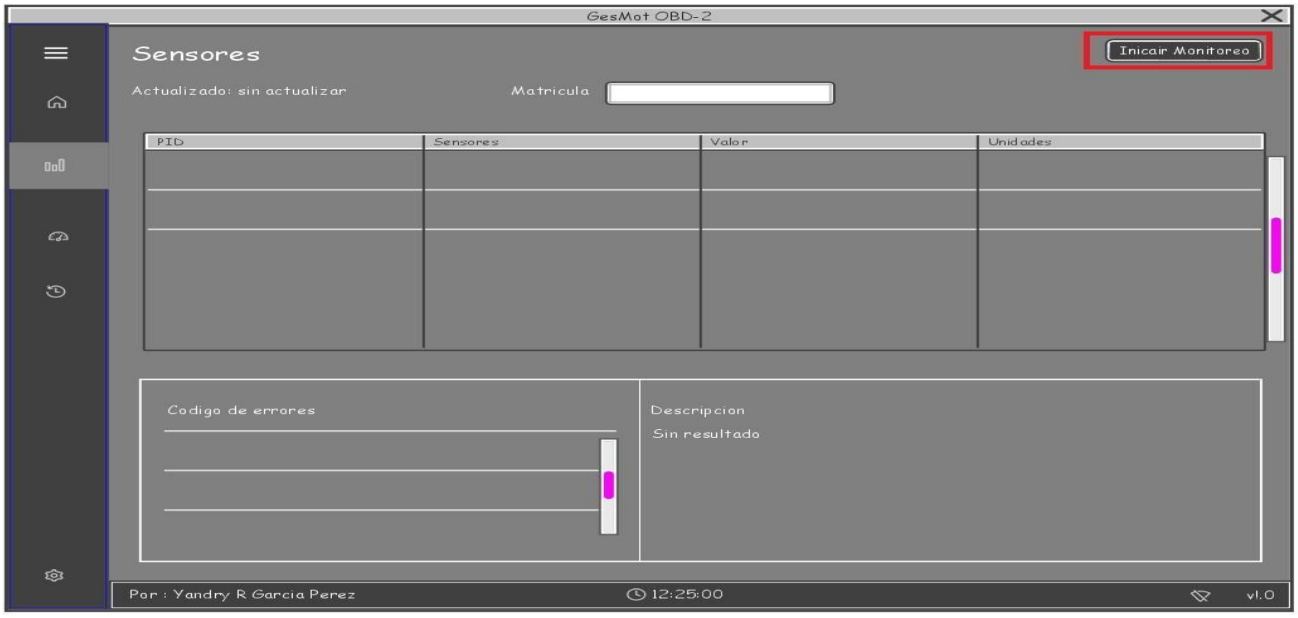


Tabla 22 HU Gestionar datos adquiridos (Fuente: Elaboración propia).

Historia de usuario	
Número: 3	**Número del requisito:** Salvar datos adquiridos
Programador: Yandry R García Pérez	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 130 h

Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 130 h

Descripción:

- **Objetivo:** Almacenar los datos de los sensores del vehículo por un periodo de 31 días. Visualizarlos y Graficarlos
- **Precondiciones:** Se debe tener conectado el dispositivo al computador y adquiridos datos.
- **Acciones a realizar:**

Salvar:

Al introducir los datos de la matrícula y dar clic en el **botón guardar** se almacenarán los datos de los sensores y códigos de fallos del vehículo.

Visualizar:

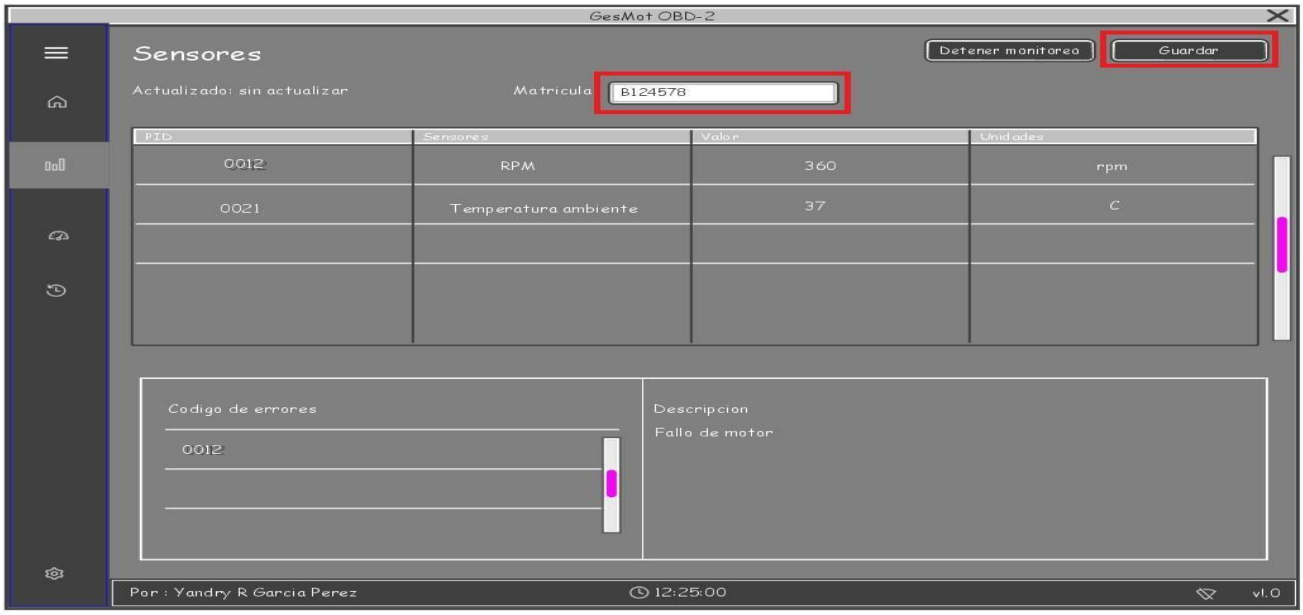
Al **seleccionar la chapa** de un vehículo registrado se da clic en el **botón actualizar** y se visualiza los datos almacenados de los sensores y las fallas del vehículo. Al dar **doble clic** sobre los **códigos de fallo** se da una descripción del código

Graficar:

Al **seleccionar la chapa de un vehículo** registrado se da clic en el **botón actualizar** y se visualizan los datos almacenados de los sensores. Se **selecciona el sensor** a analizar y se da clic en el **botón graficar**.

Observación:

Prototipo de interfaz:



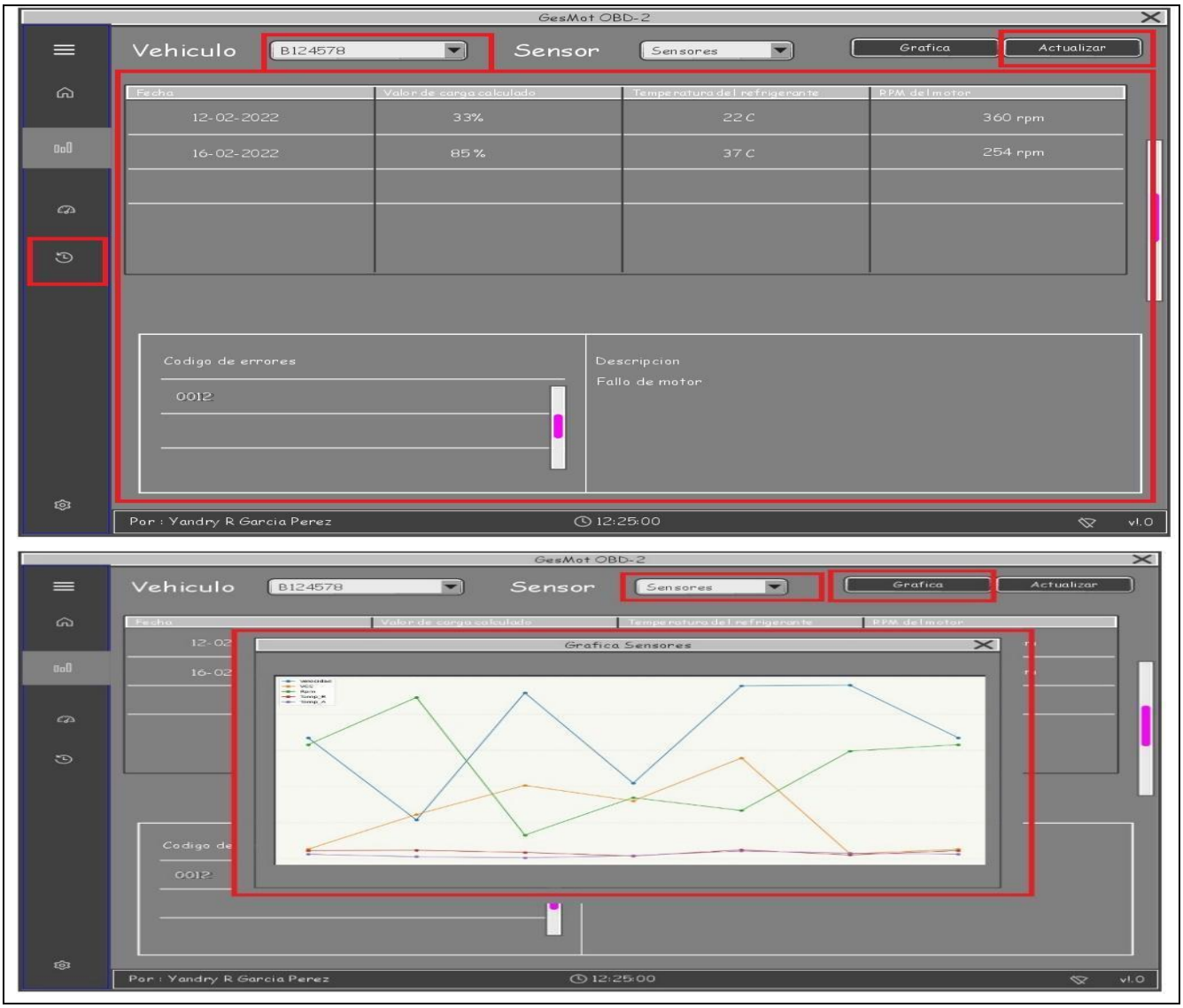


Tabla 23 HU Visualizar datos en tiempo real del vehículo (Fuente: Elaboración propia).

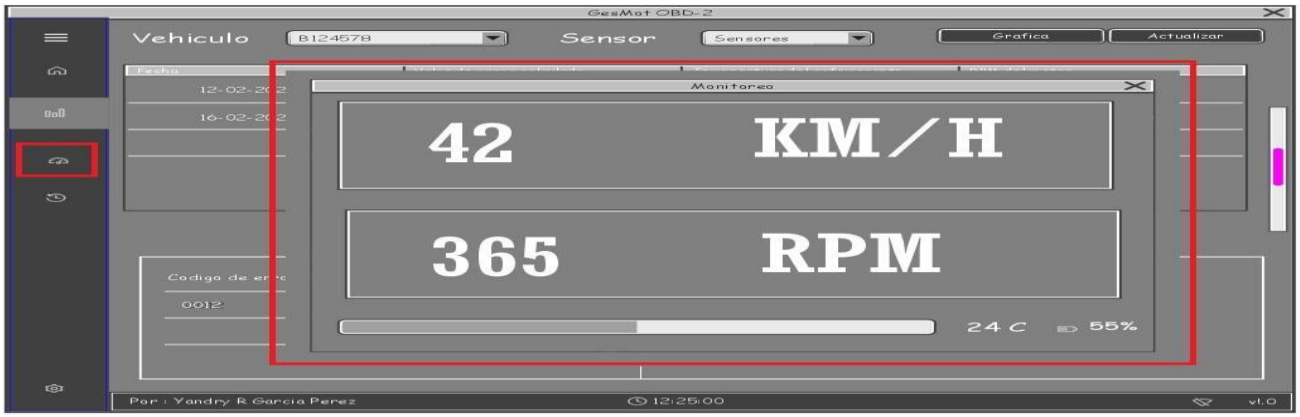
Historia de usuario	
Número: 4	**Número del requisito:** Visualizar datos en tiempo real del vehículo
Programador: Yandry R García Pérez	**Iteración asignada:** 1
Prioridad: Baja	**Tiempo estimado:** 72 h
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 72 h

Descripción:

- **Objetivo:** Visualizar datos en tiempo real del vehículo, mostrando los valores en el momento instante de las Rpm, Velocidad, Batería y Temperatura del Motor.
- **Precondiciones:** Se debe tener conectado el dispositivo al computador y adquiriendo datos.
- **Acciones a realizar:** Una vez adquiriendo datos se da clic en el botón monitor, en el panel lateral y aparecerá una ventana con esta información.

Observación:

Prototipo de interfaz:



Anexo 4 Interfaces Graficas de usuario

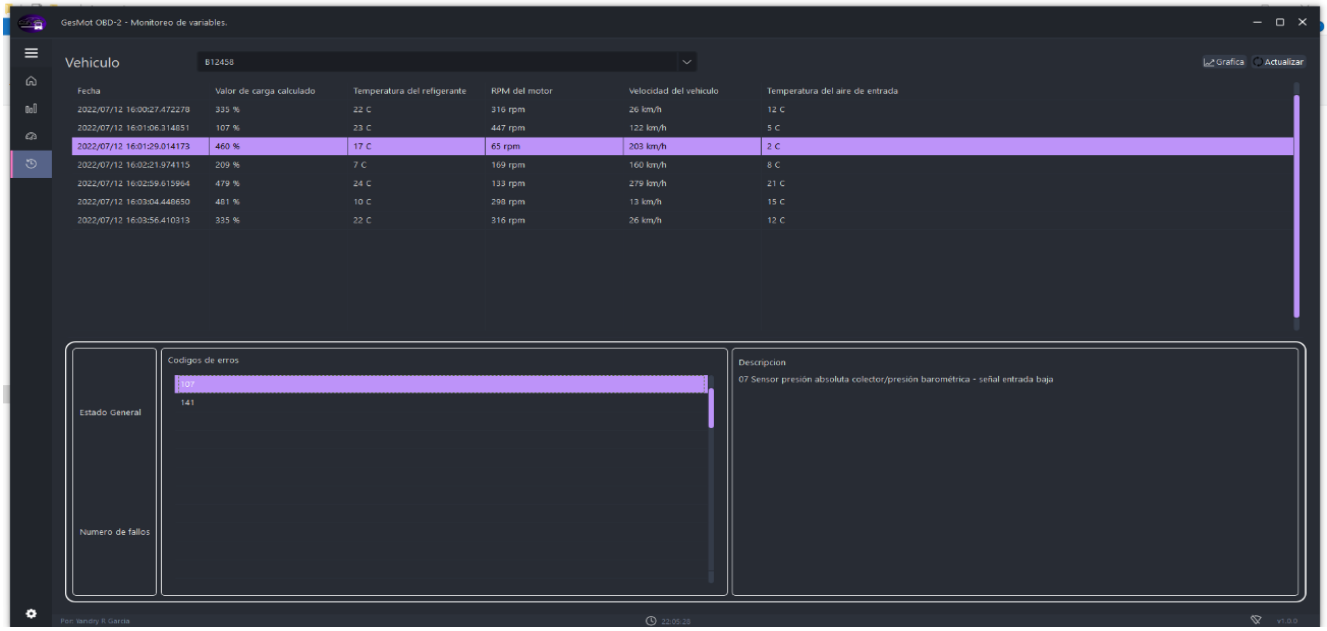


Figura 18 Interfaz gráfica, Estadística (Fuente: Elaboración propia).

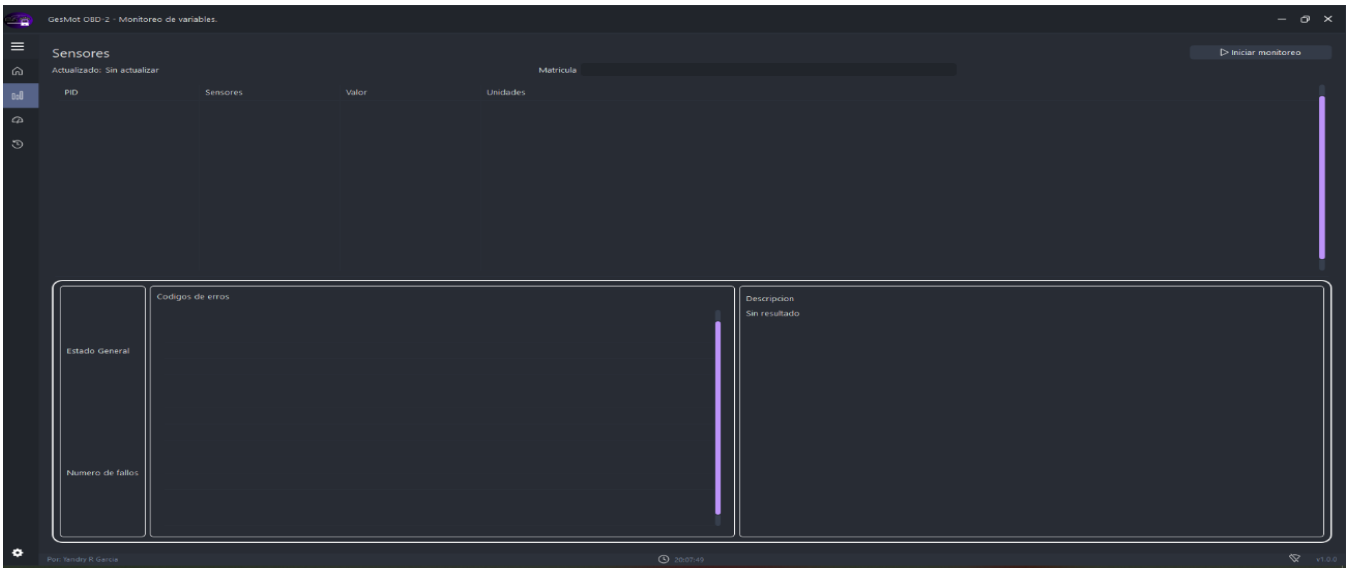


Figura 19 Interfaz gráfica, Diagnostico (Fuente: Elaboración propia).

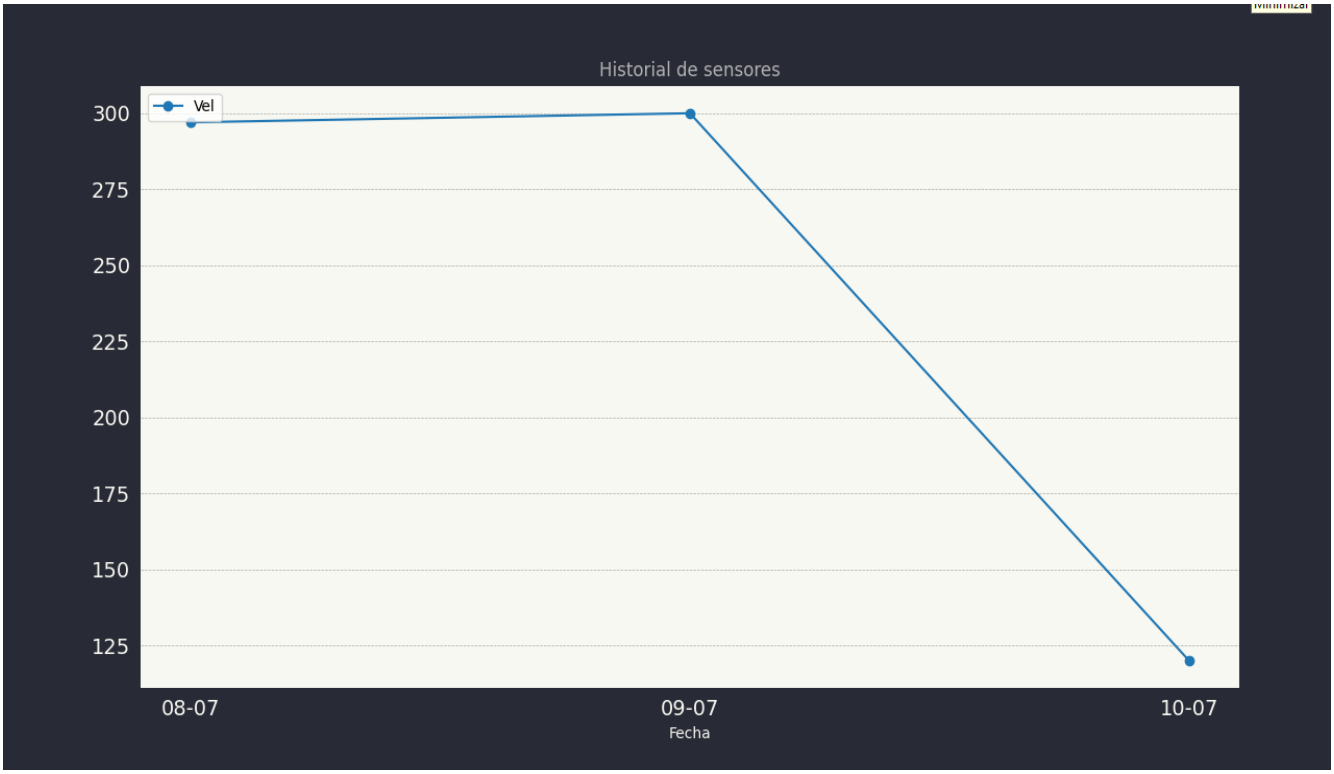


Figura 20 Interfaz gráfica, Gráfica sensores (Fuente: Elaboración propia).

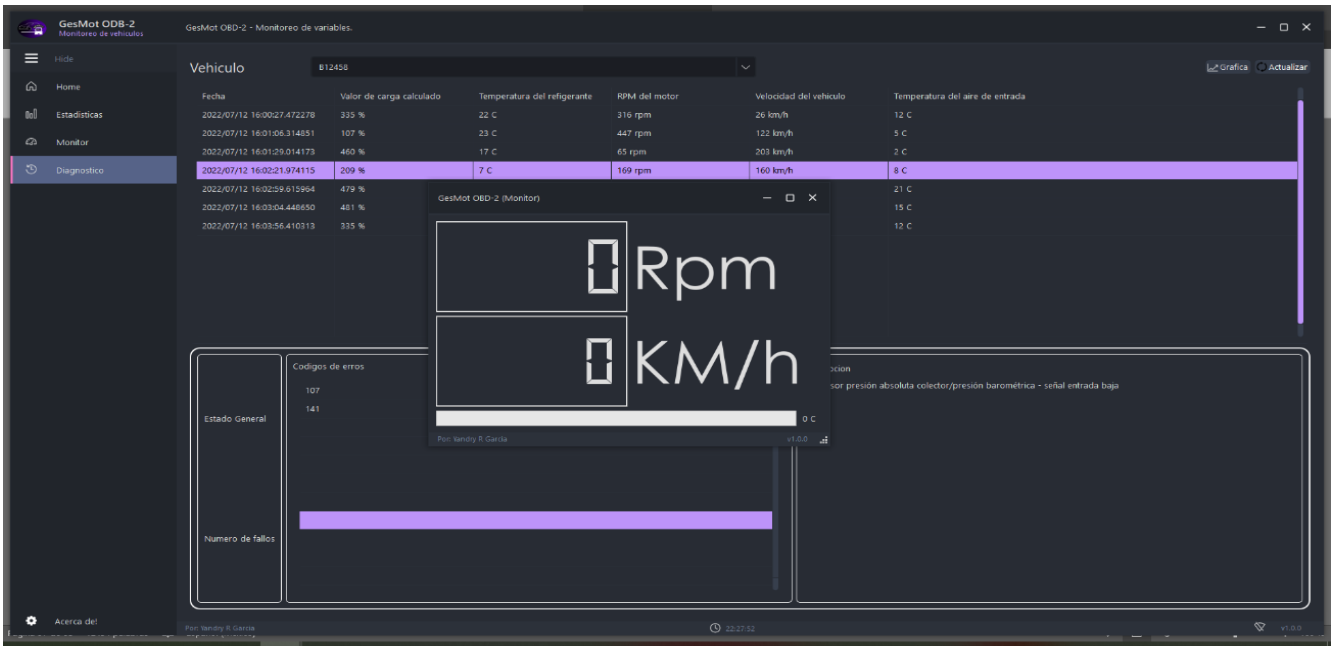


Figura 21 Interfaz gráfica, Estadísticas (Fuente: Elaboración propia)

Anexo 5 Pruebas de Aceptación por Historia de Usuario

***Descripción General*:** El Sistema debe Gestionar la adquisición de datos, para poder iniciar adquisición, salvar datos y visualizarlos

Condiciones de ejecución: *Debe estar un Arduino conectado por puerto USB, e inicializada la conexión.*

Tabla 24 Prueba de aceptación Historia de usuario 2 (Fuente: Elaboración propia)

Escenario	Descripción	Matricula	Datos	Respuesta del sistema	Flujo central
Los datos a salvar son correctos	El sistema debe detectar que los datos adquiridos son correctos para salvar	B12012	Datos del Arduino	Se notifica que fueron guardados	Se debe inicializar la adquisición, introducir la matrícula, y presionar el botón guardar
Los a salvar son incorrectos	El sistema no valida que la matrícula es correcta	123	Datos del Arduino	Se notifica que la matrícula es	

				incorrecta	
	El sistema valida que la matrícula es vacía		Datos del Arduino	Se notifica que la matrícula está vacía	