FACULTAD 1

Título:

Sistema para gestionar el diagnóstico en las flotas de vehículos mediante el dispositivo OBD2.

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

Autor(es): Amanda Ojeda Fernández.
Tutor(es): Ing Vladimir Campos Kindelan.
Ing Julio Alberto Leiva Durán.
Co-tutor: Ing. Osberto Prieto Pérez.

La Habana, abril de 2022

¨Año 64 de la Revolución¨

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título "**Sistema para gestionar el diagnóstico en las flotas de 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 únicos autores de su contenido. Para que así conste firma la presente a los 7 días del mes de diciembre del año 2022.

<nombre del autor>

_____
Firma del Autor

<nombre del tutor>

_____
Firma del Tutor

<nombre del autor>

_____
Firma del Autor

<nombre del tutor>

_____
Firma del Tutor

DATOS DE CONTACTO

Autor:

Amanda Ojeda Fernández

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: amandaof@estudiantes.uci.cu

Tutor(es):

Ing. Julio Alberto Leyva Durán.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jaleyva@uci.cu

Ing Vladimir Campos Kindelan

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: vladimirc@uci.cu

Cotutor(es):

Ing. Osberto Prieto Pérez.

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: oprietop@uci.cu

AGRADECIMIENTOS

A toda mi familia, en especial a mi papá que se ha sacrificado y me ha enseñado a luchar para conseguir mis metas y ha luchado por mis sueños, a mi mamá que ha dado todo su empeño porque yo logre mis sueños, y que siempre ha estado ahí para mí, sin importar mis decisiones, mis hermanos por su dedicación, cariño y apoyo, en especial a mi hermano Luisito y a mi cuñada Lía. A mi novio y su tía por sus consejos, su compañía, su dedicación y por quererme tanto en esta etapa tan importante. A mis tutores, Osberto, Vladimir y Julio, por todo el apoyo dado y por mostrar total disposición cada vez que necesité de ellos, por dedicarme su tiempo y paciencia. A Beatriz que ha sido compañera, amiga y hermana siempre, no hay palabras que describan cuan agradecida estoy por todos los momentos compartidos. A todos mis amigos desde el barrio hasta la universidad, que más que amigos son mis hermanos a Leisbel, Pedro, Nélida, Drake, Lobaina, Alejandro Santana, Juan Ramón, en fin, a todos los niños de mi aula y las niñas de mi apto. El tiempo sigue pasando y siempre han estado ahí cerca de mi ofreciéndome lo mejor que tienen. Les agradezco por toda su ayuda incondicional, sin importar la hora ni el momento que la necesitara, por todo su apoyo y los consejos brindados en el transcurso de este trabajo. A todos mis profesores que me han ofrecido lo mejor de ellos en cada clase y hasta en cada regaño. A todas las personas, que de una forma u otra aportaron un granito de arena a lo largo de mis estudios e hicieron posible el desarrollo de este trabajo.

DEDICATORIA

Quiero dedicar con todo mi corazón esta tesis a mis padres Favien y Daniel por haberme forjado como la persona que soy en la actualidad; muchos de mis logros se los debo a ustedes entre los que se incluye este. Me formaron con reglas y con algunas libertades, pero al final de cuentas, me motivaron constantemente para alcanzar mis sueños. A mi hermano Luisito no solo por estar presente aportando buenas cosas a mi vida, sino por los grandes momentos de felicidad y de diversas emociones que siempre me ha causado.

Muchas gracias, los amo.

RESUMEN

Una adecuada planificación y control del mantenimiento y del diagnóstico técnico de vehículos, como apoyo a la toma de decisiones, en las empresas de transporte dará lugar a aprovechar de manera más rentable la explotación del transporte y a la economía de una empresa. La llegada al mercado de dispositivos capaces de leer información del vehículo a través del sistema de diagnóstico a bordo (OBD) permite obtener datos provenientes de sensores de distintos módulos del vehículo hasta códigos con la indicación de una avería detectada. Este proyecto se enmarca en la creación de un sistema que gestione el diagnóstico en las flotas de vehículos a través del dispositivo OBD2 y la facilidad de realizar las búsquedas de fallas en cualquier tipo de vehículo. La propuesta de solución fue desarrollada mediante la metodología Proceso Unificado Ágil (AUP-UCI) en su escenario cuatro, Visual Paradigm para modelar los diagramas UML, el framework Flask para el desarrollo de la aplicación Web, con el cual es usado el lenguaje de programación Python y Visual Studio Code como editor de código, MySQL como gestor de base de Datos. Finalmente se ha realizado un estudio de diferentes pruebas para validar el correcto funcionamiento de la aplicación.

PALABRAS CLAVE

Mantenimiento, Diagnóstico, Falla, Dispositivo, Vehículo

ABSTRACT

An adequate planning and control of the maintenance and technical diagnosis of vehicles, as a support for decision-making, in transport companies will lead to more profitable use of transport and the economy of a company. The arrival on the market of devices capable of reading vehicle information through the on-board diagnostics (OBD) system allows obtaining data from sensors of different vehicle modules up to codes with the indication of a detected fault. This project is part of the creation of a system that manages the diagnosis in vehicle fleets through the OBD2 device and the ease of performing fault searches in any type of vehicle. The solution proposal was developed using the Agile Unified Process (AUP-UCI) methodology in its fourth scenario, Visual Paradigm to model the UML diagrams, the Flask framework for the development of the Web application, with which the programming language is used. Python and Visual Studio Code as code editor, MySQL as database manager. Finally, a study of different tests has been carried out to validate the correct functioning of the application.

KEYWORDS

Maintenance*, Diagnostic, Failure, Device, Vehicle*

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

OPINIÓN DEL(OS) TUTOR(ES)

AVAL DEL CLIENTE

INTRODUCCIÓN

Desde la revolución de las máquinas las empresas comercializadoras de automóviles sienten la necesidad de seguir desarrollando más comodidad y control sobre los vehículos. Antiguamente, el gestor de flota dependía mucho de la información que le aportasen los conductores de los vehículos, ya fuera por llamadas, informes, etc. Esto tenía dos inconvenientes: la dependencia de la capacidad y honestidad de los trabajadores a la hora de realizar esos informes y el tiempo de gestión que suponía a estos y al gestor rellenar, trasladar a un programa de gestión e interpretar los datos recogidos (Ramos Coria, 2014).

Para gestionar los datos de dichos procesos que se encargan de suministrar los recursos necesarios para mejorar los productos y servicios de una organización surge como principio la toma de decisiones, paso muy importante para enfrentarnos a una situación concreta. Además, es el resultado del análisis de varias alternativas de decisión para tomar un dictamen (Yunier Rodríguez CruzI; María Pinto MolinaII,2010).

Finalmente, en los años 80, se avanza en el empleo de las Tecnologías de la Información y la Comunicación (TIC), surge el ordenador de a bordo en los vehículos, y, en breve espacio de tiempo, se hace viable su conexión a diversos satélites y a las redes inalámbricas terrestres (Ramos Coria, 2014).

Uno de estos sistemas es el Sistema OBD-II que se comenzó a utilizar de forma obligatoria en los nuevos automóviles en los Estados Unidos de América desde 1996, OBD-II detecta algún tipo de problema a través de los transductores, se enciende una luz de advertencia en el tablero, que alerta al conductor de la falla existente. Después pasa a guardar la información sobre las fallas detectadas en la memoria de la computadora del automóvil, lo que facilita al técnico automotriz encontrar los problemas para corregirlos posteriormente (Ramos Coria 2014).

Por otra parte, en las flotas de transporte, el combustible tiene especial relevancia en su estructura de costes, y más aún con los actuales precios a los que se cotiza el crudo en el mercado. Para el correcto desarrollo de su actividad económica, se hace necesaria la realización de una planificación y control eficiente de combustible en las mismas, ya que puede ocurrir alguna falla en un vehículo y por motivo de esta llegar al consumo innecesario de combustibles (IDAE, 2006).

Los precios internacionales de petróleo Brent y WTI encerraron la quincena de febrero de 2022 con una cotización de USD/barril 94,3 y 93,3, respectivamente, las mayores desde octubre de 2014 y incrementándose cerca de 70% interanualmente.

La cotización internacional del crudo impacta en los precios de combustibles provocando que los países importadores netos deban realizar ajustes en el precio al consumidor (Hugo Roig).

En Cuba se le ha concedido prioridad máxima a la socialización de las Tecnologías de la Informática y las Comunicaciones (TIC). Cuando se hace referencia a la informatización de la sociedad cubana, se está hablando de la aplicación ordenada y masiva de las tecnologías de la informática, para mediante su uso racional y adecuado lograr una mayor generación de riquezas. (Periódico Granma, 2014).

Cinco pilares rigen el proceso de la informatización de la sociedad en la isla: ciberseguridad, infraestructura, creación de contenidos, marco legal y cultura responsable; todos en una carrera de enormes retos, pero certeros y necesarios para alcanzar un mayor desarrollo económico y social (Marine,2020).

Datos del Ministerio de Comunicaciones de la isla (Mincom) refieren que actualmente siete millones de cubanos acceden a Internet por diferentes vías, de ellos 4, 2 lo hacen a través de datos móviles.

De estos últimos, 1,3 millones trabajan con tecnología de cuarta generación de telefonía celular (4G LTE) (Tomado de Prensa Latina, 2020).

Uno de los sectores que se inclina a informatizar sus procesos es el sector del transporte, para facilitar el manejo de la información de fallos y tener un mejor control sobre el consumo de combustible. Ejemplo es la revisión técnica automotor, que no es más que las plantas conocidas como ¨somatón¨, el diagnóstico se realiza con el objetivo de obtener el certificado que un vehículo es apto para circular por la vía (Ramadán Arcos, 2015).

En lo que va de año han sido retiradas más de 700 licencias operativas y cerca de 1 200 certificados de revisión, según el ingeniero Erasmo Arias Verdecia, director de Inspección Estatal del Transporte en la capital, por fallos en los vehículos y en espera de una toma de decisión por parte de los analizadores (Ramadán Arcos, 2015).

La tarea de planificar y controlar el consumo de combustible necesita ser realizada de forma estructurada y supervisada. Una adecuada planificación y control de combustible es necesario en las empresas de transporte puesto que dará lugar a aprovechar de la manera más rentable cada litro de combustible adquirido, contribuyendo con ello a la economía de la empresa. Además, contribuye a una mayor eficiencia energética en la realización de sus servicios, mejorando la eficiencia de cada vehículo, a través del control y seguimiento individualizado de los mismos (IDAE, 2006).

La Universidad de las Ciencias Informáticas (UCI) es un ejemplo y fruto del desarrollo de las TIC contando con excelentes tecnologías, centro referenciado nacionalmente en cuanto a la calidad de software siendo, además, un pilar fundamental de la estrategia del Comandante en Jefe Fidel Castro Ruz la cual es: “graduar profesionales altamente capacitados y cada vez más comprometidos con la

Revolución", capaces de afrontar las emergentes tecnologías que se van incorporando a Cuba. La Universidad tiene, entre sus principales objetivos, la producción de software y servicios informáticos a través de su modelo de formación que incluye la vinculación estudio-trabajo (Espinosa 2015; Pérez 2015).

Dentro de la estructura de la universidad se encuentra la Xetid, empresa líder de desarrollo de soluciones tecnológicas para la informatización de la sociedad. La empresa está dedicada a la industria del software, la automática y las comunicaciones, asume como misión la proyección, diseño, desarrollo y comercialización de productos y servicios a partir del uso de las TIC (Xetid,2022).

Actualmente la empresa Xetid no cuenta con ningún sistema que proporcione las funcionalidades requeridas para darle cumplimiento a todas las necesidades demandadas para la gestión del diagnóstico de los vehículos. No consta de ningún sistema que facilite una posible solución o premedite los fallos de un vehículo en algún momento determinado. Hay ausencia de un registro para llevar el control y planificación del mantenimiento de los autos, así como, la inexistencia de varios tipos de mantenimientos para los vehículos, provocando el deterioro de vehículos, el desorden del tiempo determinado de revisión para cada auto y el consumo innecesario de combustible. Además, en los últimos 3 años, ha superado los $ 100,000,000 cup entre el pago de turnos en el Somatón, la tardanza entre las inspecciones provocando gastos por mantenimiento elevados los cuales son causa a la falta de la gestión del diagnóstico y control del mantenimiento de sus vehículos.

A partir de estos elementos, se plantea el siguiente **problema de investigación:**

¿Cómo gestionar el diagnostico técnico de los vehículos para la planificación de las actividades de mantenimiento?

Para solucionar el problema planteado se declara como **objeto de estudio**: La gestión de información asociado al diagnóstico mecánico, donde el **campo de acción** estaría enmarcado en el desarrollo de los sistemas para la gestión del diagnóstico técnico de vehículos, como apoyo a la toma de decisiones.

Teniendo en cuenta el problema de investigación se define como **objetivo general:** Desarrollar un sistema que gestione el diagnóstico en las flotas de vehículos a través del dispositivo OBD2 que facilite la búsqueda de fallas en cualquier tipo de vehículo.

El objetivo general planteado será desglosado en los siguientes **objetivos específicos**:

- Evaluar aspectos teóricos-conceptuales sobre sistemas de gestión relacionados con el diagnóstico de un vehículo.
- Identificar las funcionalidades que debe tener el sistema mediante un estudio de los softwares existentes en el mundo encargado de la gestión de diagnóstico de un vehículo.

- Modelar la propuesta de solución para el sistema de gestión de diagnóstico de un vehículo mediante los artefactos ingenieriles que sugiere la metodología seleccionada.
- Implementar el código fuente.
- Validar los resultados de la implementación mediante pruebas de funcionamiento.

Una vez planteado el objetivo general, así como los específicos, se identificaron los siguientes métodos:
Los métodos científicos y técnicos de investigación utilizados en la elaboración de este trabajo fueron los siguientes:
Se utiliza la entrevista dentro de los **métodos empíricos**, para la obtención y elaboración de los datos y el conocimiento de los hechos fundamentales que caracterizan la situación actual. Se le realizó una entrevista al cliente Osberto Prieto Pérez donde expuso las necesidades que presentaba la empresa para limitar los objetivos del proyecto.
Encuesta: permitió mediante un cuestionario pre-elaborado, obtener información sobre el tiempo real que demoran los especialistas funcionales en tomar decisiones cuando se detecta una falla y a la vez se notó el descontrol del mantenimiento en los vehículos de la empresa.
La utilización del **análisis y la síntesis** es significativa como **método teórico** de ahí se hará un desglose de todo el proceso facilitando su entendimiento, permitiendo con ello descubrir sus características y relaciones más específicas. Después de un estudio más detallado de la problemática se agruparon conceptos y se realizó un orden para las actividades que se llevaran a cabo para el desarrollo del proyecto.
La inducción y la deducción se utilizan en varias fases de la investigación, donde de las ideas generales se pueden deducir unas más simples y llegar a conclusiones particulares. Se realizaron lluvias de ideas para deducir la solución más óptima a la problemática.
El método de **modelación** es necesario por el gran peso que tienen las propuestas, alternativas y estrategias en el trabajo, donde la realidad es representada a través de un modelo que guía todo el proceso. Se confeccionaron distintos diagramas como el del modelo de dominio y el modelo de base de datos.
Histórico-Lógico: permitió realizar un estudio sobre las principales tendencias de la web respecto a los softwares existentes para la gestión de un diagnóstico de las fallas de un vehículo.

Estructura:
El trabajo de diploma se estructura de la siguiente manera: introducción, tres capítulos que serán descritos a continuación, conclusiones, recomendaciones, bibliografía y anexos.

Capítulo 1: Fundamentación del sistema Diagnóstico.

Contiene la fundamentación teórica del desarrollo que permita elaborar el marco teórico referencial para la presente investigación. de un. Se realiza un estudio del estado del arte acerca los sistemas de gestión para el diagnóstico de los vehículos y de las tecnologías actuales que facilitan el desarrollo del trabajo, describiendo los principales aspectos de las herramientas, tecnologías y metodologías a utilizar relacionados con el diseño, elaboración y validación para el diagnóstico de un vehículo con la herramienta OBD2.

Capítulo 2: Implementación del sistema Diagnóstico.

En el presente capítulo se describen las principales características que debe presentar la solución propuesta, se seleccionará la herramienta necesaria para cumplir los objetivos de la presente investigación. Esta solución es basada en la definición de los requisitos y la arquitectura. Se presentan los principales artefactos propuestos en la fase de modelación en la metodología de desarrollo a aplicar y se fundamentan los patrones de diseño empleados en su elaboración.

Capítulo 3: Verificación y validación del sistema Diagnóstico.

En el presente capitulo se evalúa el grado de calidad y fiabilidad de los resultados obtenidos en el desarrollo de la presente investigación. Se muestran los resultados en cada caso luego de aplicar las técnicas de validación de los requisitos tanto al diseño del sistema como a la solución desarrollada. Para la validación del diseño se aplican métricas y para la validación de la solución se define una estrategia de prueba aplicando los 4 niveles de pruebas con sus respectivos métodos y técnicas. Por otra parte, se realiza una verificación del cumplimiento del objetivo general de la investigación.

.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA DEL SISTEMA DIAGNÓSTICO

Introducción

Contiene la fundamentación teórica del desarrollo que permita elaborar el marco teórico referencial para la presente investigación. de un. Se realiza un estudio del estado del arte acerca los sistemas de gestión para el diagnóstico de los vehículos y de las tecnologías actuales que facilitan el desarrollo del trabajo, describiendo los principales aspectos de las herramientas, tecnologías y metodologías a utilizar relacionados con el diseño, elaboración y validación para el diagnóstico de un vehículo con la herramienta OBD2.

1.1 Sistema para gestionar fallas en un vehículo

El ordenador a bordo de un vehículo es uno de los instrumentos que se usa para el diagnóstico remoto de un automóvil, por lo que el diseño y correcta confección es esencial para alcanzar los objetivos propuestos. En lugar de esperar a que el vehículo falle, los clientes pueden utilizar la telemática con un sistema de diagnóstico para determinar el estado de su vehículo. Con este enfoque pueden identificar fácilmente los vehículos que tienen problemas y mitigar las fallas de una manera controlada y programada, además de gestionar el mantenimiento de los vehículos.

Para el desarrollo de la presente investigación se exponen los siguientes conceptos del negocio para un mejor entendimiento de los procesos:

Gestión

La gestión es un conjunto de procedimientos y acciones que se llevan a cabo para lograr un determinado objetivo. La gestión de información es el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, así como para mejorar los procesos, productos y servicios de la organización. Es decir, en términos generales, la gestión es el arte de saber lo que se quiere hacer y a continuación, hacerlo de la mejor manera y por el camino más eficiente (Quiroga, Lourdes Aja,2015)

Funciones de la gestión

Los pasos de la gestión, principalmente en el ámbito empresarial, son los siguientes:

Planificación: Se fijan los objetivos a corto y largo plazo. Esto, partiendo de un análisis de la situación actual.

Organización: Se determinan los procedimientos y estrategias a seguir para conseguir los objetivos planteados.

Dirección: Es la puesta en marcha de lo planificado, teniendo en ocasiones que existir un gestor que lidere a un grupo de personas para que todos trabajen en la consecución de los mismos objetivos.

Control: Es la etapa final, cuando se contrastan los resultados obtenidos con lo planificado con antelación.

Figura 1: Funciones de la gestión. (Fuente: Elaboración propia)

Software de gestión

Es un sistema informático integrado por múltiples herramientas que individualmente se utilizan para ejecutar tareas administrativas, y que, en conjunto, simplifica los procesos operativos y productivos. Es aquel que se integra a la perfección con las actividades de tu empresa y dar servicio a las necesidades que surgen de las mismas. Se encargan de la gestión diaria y continúa, de los diferentes escenarios y procesos, que se requieren en el día a día de cualquier empresa, permitiendo su inclusión, consulta, modificación, fusión o borrado, entre otras acciones, a través de diferentes dispositivos de comunicación: móviles, tablets, ordenadores, consolas, etcétera (Pressman, R. 2010).

Diagnóstico

El diagnóstico es un procedimiento ordenado, sistemático, para conocer, para establecer de manera clara una circunstancia, a partir de observaciones y datos concretos. El diagnóstico conlleva siempre una evaluación, con valoración de acciones en relación con objetivos. El término incluye en su raíz el vocablo griego ‘gnosis’, que significa conocimiento (https://concepto.de/diagnostico/#ixzz7ksqPkoXE).

La definición de diagnóstico de Andrade de Souza: «Un método de conocimiento y análisis del desempeño de una empresa o institución, interna y externamente, de modo que pueda facilitar la toma de decisiones «

1.2 Caracterización de OBD2

OBD (*On Board Diagnostics*) es un sistema de diagnóstico a bordo en vehículos (automóviles, autobuses y camiones). Actualmente se emplean los estándares OBD-2 que aportan un monitoreo y control completo del motor y otros dispositivos del vehículo. El OBD no tuvo lugar hasta que los autos estuvieron equipados con controles computarizados. La compañía General Motors implemento una versión primeriza de OBD en algunos de sus vehículos de 1980. El objetivo del sistema OBD original era promover el aire limpio por medio de asegurar que los componentes del control de emisiones siguieran funcionando (Israel Cervantes Alonso & Saúl Osbor'n Espinosa Solís, 2010).

OBD-II

OBD II es la segunda generación del sistema de diagnóstico a bordo, sucesor de OBD I. Alerta al conductor cuando el nivel de las emisiones es 1.5 mayor a las diseñadas. A diferencia de OBD I, OBD II detecta fallos eléctricos, químicos y mecánicos que pueden afectar al nivel de emisiones del vehículo. Con OBD II, los dos sensores de oxígeno, uno antes y el otro después del catalizador, garantizan el buen estado químico del mismo (Tenorio Córdova & Coronel Magallanes, 2020).

El sistema verifica el estado de todos los sensores involucrados en las emisiones, como por ejemplo la inyección o la entrada de aire al motor. Cuando algo falla, el sistema se encarga automáticamente de informar al conductor encendiendo una luz indicadora de fallo (*Malfunction Indication Lamp* (MIL), también conocida como *Check Engine* o *Service Engine Soon*) (Anastasio & Enrique, 2013).

Para ofrecer la máxima información posible para el mecánico, guarda un registro del fallo y las condiciones en que ocurrió. Cada fallo tiene un código asignado. El mecánico puede leer los registros con un dispositivo que envía comandos al sistema OBD II llamados PID (Parameter ID) (Anastasio & Enrique, 2013).

Actualmente se puede conectar con la máquina de diagnosis de diferentes maneras, mediante Bluetooth, WiFi, USB, cayendo en desuso el protocolo de conexión por el puerto serie (RS232). Este enlace, unido a un software ejecutándose desde un ordenador o un terminal móvil permite la monitorización en tiempo real de códigos de error y diversos parámetros directamente de la centralita del motor tales como las revoluciones del motor, el consumo de combustible en tiempo real (sin que el automóvil lleve equipado ordenador de abordo) o la temperatura del aceite, entre muchos otros parámetros dependiendo del modelo. El controlador ELM327 es el más extendido para establecer dichos enlaces entre la centralita del motor y el dispositivo con el software instalado (Tenorio Córdova & Coronel Magallanes, 2020).

Características del protocolo OBD2 (Tenorio Córdova & Coronel Magallanes, 2020)

- El sistema OBD2 almacena un registro de las fallas encontradas en el vehículo y en qué condiciones ocurrieron.
- Por lo general el puerto OBD2 se localiza en la consola central del carro o a un lado del asiento del copiloto.
- Las normativas de OBD2 son aplicables a vehículos alimentados por gasolina y gasoil (diésel) o algún otro combustible alternativo.
- Se puede hacer la conexión con la máquina de diagnosis de varias maneras: a través de WiFi, bluetooth, cable USB y utilizando un software de instalación mediante un ordenador o dispositivo móvil.

Componentes del sistema OBD

Los componentes del sistema OBD-II son: la ECU, 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*], ubicada 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 (Tenorio Córdova & Coronel Magallanes, 2020).

Figura 2: Dispositivo OBD2. (Fuente: Elaboración propia)

1.3 Estudio de sistemas homólogos

1- Dingsunbao 2.0

Los propietarios de automóviles pueden usar sus teléfonos inteligentes para capturar videoclips de sus automóviles, siguiendo las pautas en pantalla. La información sobre daños al vehículo se mostrará automáticamente, incluido dónde y cómo reparar el vehículo. Incluye 46 tecnologías patentadas, como

localización y mapeo simultáneos, un modelo móvil de aprendizaje profundo, detección de daños con transmisión de video, visualización de resultados con realidad aumentada y otros.

2- ScanTool de AutoEnginuity

Es un excelente software de diagnóstico automotriz para tu computadora con Windows. Los propietarios de ScanTool de AutoEnginuity afirman que la cobertura es la característica principal de la aplicación. La aplicación admite opciones de cobertura para 48 fabricantes de automóviles. Tiene excelentes interfaces fáciles de usar sin sacrificar su facilidad de uso y puede desplazarse, permite hacer zoom y muestra varios gráficos en un gráfico.

La aplicación permite mostrar datos en una forma cómoda en lugar de datos sin procesar. Los datos se registran en dos formatos: XML para usar en navegadores y CSV para hojas de cálculo con la capacidad de cambiar el formato y ver registros sin conexión.

Datos de sensores personalizables. Se puede cambiar la forma en que muestra los datos sobre la frecuencia de muestreo, los rangos, los puntos de activación de audio, las unidades y el valor de escala del sensor.

3- ProScan

Se considera uno de los escáneres OBD basados en PC más fáciles de usar. ProScan a menudo se vende como una mezcla con hardware y software y contiene todos los cables, equipos y software necesarios para convertir cualquier PC en un lector de códigos OBD2.

Administración de conexión de vehículos. El *Vehicle Connection Manager* es una interfaz que ve al iniciar el programa. Para realizar la conexión entre PC y vehículo, el usuario debe seleccionar el perfil del vehículo y conectarse a través del programa. Mostrará el estado de la conexión y verá el informe si ocurre la falla.

Generador de informes de diagnóstico. ProScan permite generar informes de diagnóstico para un automóvil con un solo botón. Revise el detector de luz del motor. Cuando aparece la luz de verificación del motor de un automóvil, un vehículo guardará un código que reconoce el problema detectado (DTC). Y la herramienta proporcionará una lista de dichos DTC guardados con su descripción para que el usuario pueda detectar el problema.

4- EOBD Facile

Los usuarios pueden hacer un diagnóstico de un vehículo usando Mac OS y averiguar el motivo de los indicadores del motor sin el paquete del fabricante. La conexión es simple y luego el usuario puede diagnosticar y ver los resultados en tiempo real.

Las funciones principales de la programación del software EOBD *Facile* OBD son:

- Ver fallas del motor de códigos y descubrir su significado.
- Limpiar la luz de verificación del motor.
- Supervisar códigos de error especiales del fabricante.
- Realiza grabaciones de viajes y guárdelas a través de iPhone / iPad, recuperalas más tarde utilizando la aplicación EOBD Facile.
- Crea y ve registros de datos GPS.

5- Sistema basado en redes neuronales para la detección de fallas

Se pretende que las emisiones contaminantes del vehículo sean las entradas del sistema y las salidas sean algunas de las fallas del motor. Después, mediante el proceso inverso al aplicar redes neuronales, se determinan las condiciones del motor en función de la lectura de emisiones contaminantes. Cabe mencionar que, el sistema de diagnóstico se desarrolla para tres motores.

CUBA

6- SIGREX

Para mantener la flota actualizada, se precisa previamente la actualización de clasificadores generales como son marcas, modelos y tipos de combustibles. Se gestionan los autos en explotación, de baja y, en general el estado de operación. Por otra parte, también se tiene un control de los motores de los autos, los mantenimientos, los Contratos Leasing (arrendamiento financiero) y las licencias otorgadas por la Empresa de Administración Vial y Diagnóstico Automotor (FICAV1) perteneciente al Ministerio del Transporte.

Este sistema acumula doce años de explotación continua y está diseñado como una aplicación escritorio con una arquitectura cliente-servidor. Como servidor de base de datos se utiliza el Sistema Gestor de Base de Datos Microsoft SQL Server 2000 y se utiliza el tipo de base de datos descentralizada. La aplicación cliente está desarrollada sobre la herramienta Visual Estudio 2003 y lenguaje de programación C#.

Tabla 1: Sistemas Homólogos. (Fuente: Elaboración Propia).

Sistemas Homólogos	**Completitud Funcional**	**Autenticidad**	**Reusabilidad.**	**Interopera-bilidad**	**Usabilidad**	**Costo adquisitivo**
Dingsunbao 2.0	no	no	no	si	No	Si
ScanTool de AutoEnginuity	no	no	no	si	Si	Si
ProScan	si	no	no	si	Si	Si
EOBD Facile	no	no	no	no	Si	Si
Sistema basado en redes neuro-nales para la de-tección de fallas	si	si	si	si	No	No
SIGREX	no	si	no	si	No	---

- **Completitud funcional:** Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.
- **Interoperabilidad:** Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.
- **Usabilidad:** Capacidad para ser usado. Capacidad del producto que permite al usuario operarlo y controlarlo con facilidad.
- **Autenticidad:** Capacidad de demostrar la identidad de un sujeto o un recurso.
- **Reusabilidad:** Capacidad de un activo que permite que sea utilizado en más de un sistema software o en la construcción de otros activos (ISO 25010).

Al realizar un estudio en el ámbito nacional e internacional de sistemas que se dedican a la Gestión de la información del diagnóstico de los vehículos(Tabla 1), así como la planificación y control de combustible se concluye que: los sistemas a nivel internacional integran las tecnologías más avanzadas y utilizan disímiles funcionalidades que dan respuesta a complejas exigencias, pero no cumplen con los requerimientos necesarios para gestionar las fallas de vehículos y del consumo de combustible en la empresa Xetid. Además, estos sistemas en su mayoría son privativos y su aplicación en Cuba resulta engorrosa a partir de los altos costos por concepto de licencia y soporte de las tecnologías que utilizan.

Aunque se detectaron estos percances, estas plataformas aportaron un grupo de características que se tomarán como referencia de buenas prácticas en el desarrollo de la solución propuesta con el objetivo de mejorar la aplicación, identificando requisitos funcionales y no funcionales, así como la necesidad de la accesibilidad y compatibilidad desde cualquier navegador web, independiente del sistema operativo utilizado.

1.4 Tendencias en el desarrollo de aplicaciones web

Podemos entender como desarrollo web a las acciones de diseñar, construir y mantener plataformas (aplicaciones, sitios, portales) web para entidades, organizaciones o empresas. Este proceso está compuesto por dos capas, la capa de presentación o lado del cliente *(Frontend*) y la capa de acceso a dato (*Backend*) (JA Ibarra et al 2021).

El *frontend* trabaja la interfaz visual, y permite que el usuario pueda interactuar con la aplicación o sistema. Está orientado al lenguaje de marcas y al lenguaje de programación web de ejecución en equipos clientes(JA Ibarra et al. 2021).

Se denomina *backend* a la capa de acceso a los datos de un software que no es accesible para el usuario final. Además, esta capa contiene toda la lógica de la aplicación que maneja los datos. Cabe señalar que los datos de una aplicación se encuentran almacenados en una base de datos dentro de un servidor(JA Ibarra et al. 2021).

En el contexto del desarrollo web, el *Frontend* lo conforman todas aquellas tecnologías que se ejecutan del lado del cliente, o sea, que corren en el navegador web, generalizándose en tres lenguajes HTML, CSS y JavaScript. En el backend, compuesto por las tecnologías que utilizan los servidores encontramos lenguajes de gran uso actualmente como son: Java, C#, PHP y JavaScript; además, es el Backend el que interactúa con las bases de datos, que entre los gestores más conocidos podemos mencionar Oracle, MySQL, PostgreSQL, MongoDB, entre otros (L Orozco Zarzosa, 2018).

La globalización y la constante evolución de los sistemas han permitido que la accesibilidad a las diversas herramientas informáticas que utilizamos en nuestras labores pueda llevarse a cabo tan solo con una computadora, generando así el desarrollo de numerosos lenguajes con múltiples opciones para cada usuario y de acuerdo a sus preferencias; así mismo se han creado varios framework (herramientas de trabajo) que facilitan y reducen el tiempo en la ejecución de las tareas. Entre los framework más conocidos se hallan Angular, Symfony, Node JS, Vue JS, Quásar, entre muchos otros de gran uso en nuestros días. Otra tendencia es el uso de complementos y bibliotecas de JavaScript, JQuery (que aporta facilidades para efectos, galerías de fotos dinámicas, calendarios, entre otras

prestaciones) y Bootstrap que permite adaptar el diseño a varios dispositivos y se le asocia con estilos o plantillas predefinidas fáciles de configurar (Neftalí, 2019).

1.5 Metodología de desarrollo de software

Las metodologías de desarrollo de software se utilizan para administrar o gestionar un proyecto de manera organizada y sistemática para que resulte exitoso. Ésta queda comprendida desde la fase en que se idea él proyecto hasta que se despliega (T. R. Sánchez, 2015).

La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP (Proceso Ágil Unificado), con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la universidad. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, nombrada Ejecución y agregándose también una nueva fase llamada Cierre (Rodriguez, Vazquez, Aliaga 2012).

Las 3 fases son:

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.

La metodología de software **AUP-UCI** a partir del modelado de negocio propone tres variantes a utilizar en los proyectos, como son: CUN (Casos de uso del negocio), DPN (Descripción de proceso de negocio) o MC (Modelo conceptual) y existen tres formas de encapsular los requisitos los cuales son: CUS (Casos de uso del sistema), HU (Historias de usuario), DRP (Descripción de requisitos por proceso), surgen cuatro escenarios para modelar el sistema en los proyectos, en esta investigación se estará trabajando sobre el escenario No 4:

Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU (Sánchez, 2015).

A partir del análisis e investigación realizada, se utilizó la variante (HU) y el escenario #4 de la metodología AUP-UCI, debido a que dicha metodología es apropiada para proyectos pequeños permitiendo disminuir las probabilidades de fracaso, por ser un producto de fácil uso utilizando cualquier herramienta. Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software.

1.6 Herramientas y tecnologías

Las herramientas CASE (Ingeniería de Software Asistida por Computadora) son un conjunto de ayudas que permiten el desarrollo de programas informáticos, ayudando en su planificación, su análisis, diseño, la generación del código del software y hasta en su documentación (Mendez, 2018).

Visual Paradigmv10.0

Visual Paradigm es una herramienta de UML que es capaz de soportar el ciclo de vida de todas las fases del desarrollo de un software. Posibilita una rápida construcción de aplicaciones con una alta calidad, tiene incluida diversas funciones que permiten representar gráficamente todos lo diagramas de clases, código inverso, así como generar la documentación (Gaines, y otros, 2017).

Características de Visual Paradigm:

- Se obtienen productos de calidad.
- Está apto para soportar aplicaciones web.
- Las imágenes y reportes generados, no son de muy buena calidad.
- Tiene incluido varios idiomas.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.
- Visual Studio Codev1.56.2

Es un editor de código fuente ligero, pero de gran potencia que se ejecuta en las computadoras con sistemas operativos Windows, MacOS o Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js, y con un buen ecosistema de extensiones para otros lenguajes (En: VS Code [en línea], Disponible en: https://code.visualstudio.com [consulta: 26 de junio de 2021]).

Es seleccionado para el desarrollo de la presente investigación porque permite configurar la interfaz a nuestro gusto para poder tener un código más visible, propicia el acceso rápido a las carpetas del

proyecto y a una terminal con los detalles del problema o error. Su curva de aprendizaje es suave y existe una gran documentación al respecto.

HTML 5

El acrónimo significa lenguaje de marcado de hipertexto, se emplea para el desarrollo de aplicaciones web. Describe cosas como la sección del documento, párrafos, listas, tablas, archivos multimedia, y enlaces de hipertextos. Todos estos existen dentro de partes de un documento llamadas elemento elementos, que conforman subregiones rectangulares del navegador. HTML es un lenguaje de marcado con gramática bastante simple que el navegador puede entender. El navegador utiliza HTML para formatear y mostrar documentos. HTML5 es un estándar actual que ha sido diseñado para asegurar la compatibilidad en términos sobre la publicación sobre la Web (Bos, 2019).

El lenguaje HTML es utilizado en la propuesta de solución por ser parte fundamental en las aplicaciones web, mediante él se pueden desarrollar e integrar recursos como textos, imágenes, animaciones y otras que facilitan la construcción de la interfaz gráfica de nuestra solución.

CSS 3

Las Hojas de Estilo en Cascada (CSS por sus siglas en inglés) es un concepto que se utiliza en la informática para referirse a un lenguaje empleado en el diseño. El diseño del CSS posibilita establecer una separación entre el contenido y la forma de presentación del documento. Así se puede lograr que muchos documentos HTML compartan la apariencia, utilizando una única hoja de estilo para todos, esto permite evitar la repetición de código en la estructura (W3C, 2019).

Con el uso de CSS se logra la separación de del contenido en la interfaz de nuestra aplicación, permitiendo alcanzar el diseño de la interfaz de nuestro sistema.

JavaScript v1.8

El JavaScript es un lenguaje de programación interpretado, lo que significa que no necesita ser compilado. Orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Proviene del Java y se utiliza principalmente para la creación de páginas web. El JavaScript es una mezcla entre el Java y el HTML, es un lenguaje que se incorpora dentro de la página web, formando parte del código HTML (MDN web docs, 2020).

En nuestra aplicación nos permite aportarle el movimiento y dinamismo necesario para la ejecución y alcance de los objetivos previstos para esta, tanto del lado del servidor como en la parte del cliente.

Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE) es una aplicación que de la misma forma es un entorno de programación. En el mayor de los casos los IDEs están compuestos por en un editor de código, un compilador, un depurador y en ocasiones un constructor de interfaz gráfica de usuario.

Visual Studio Code v1.56.2

Visual Studio Code es un editor de código fuente ligero, pero de gran potencia que se ejecuta en las computadoras con sistemas operativos Windows, MacOS o Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js, y con un buen ecosistema de extensiones para otros lenguajes (Visual Studio Code 2022).

Es seleccionado para el desarrollo de la presente investigación porque permite configurar la interfaz a nuestro gusto para poder tener un código más visible, propicia el acceso rápido a las carpetas del proyecto y a una terminal con los detalles del problema o error. Su curva de aprendizaje es suave y existe una gran documentación al respecto.

Los lenguajes del lado del servidor

Los lenguajes del lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados en el lado del servidor y estos se envían al cliente en un lenguaje comprensible, no dependen del navegador a utilizar. Existen diferentes lenguajes de programación del lado del servidor. A continuación, se abordarán alguna de las características del lenguaje seleccionado para el desarrollo del proyecto, así como ventajas y desventajas.

Python v3.10.5

Python es un lenguaje que fue diseñado en 1990 por Guido Van Rossum el cual presenta diversas ventajas que lo hace ser muy atractivo a los ojos de los programadores, tanto para su uso profesional como para el aprendizaje de la programación.

Python ha sido parte importante de Google desde el principio, y lo sigue siendo a medida que el sistema crece y evoluciona. Hoy día docenas de ingenieros de Google usan Python. (Python: qué es y por qué deberías aprender a utilizarlo).

Características de Python:

- Lenguaje multipropósito.
- Lenguaje multiparadigma:
- POO.

- Programación estructurada.
- Programación funcional.
- Programación orientada a aspectos.

Es seleccionado para la codificación del proyecto ya que presenta licencia de código abierto (*Opensource*), es orientado a objetos, sencillo y rápido de programar. Se puede destacar entre una de sus ventajas que tiene una sintaxis muy legible y elegante, además de tener incluida internamente gran cantidad de funciones y librerías, por lo que ahorra tiempo y recursos.

Framework

Flask v2.2.2 es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basado en la especificación WSGI de *Werkzeug* y el motor de *templates* Jinja2 y tiene una licencia BSD, que respeta el patrón de diseño conocido como modelo–vista–controlador (MVC).

Es seleccionado el framework Flask ya que facilita el buen manejo de rutas, se considera un micro framework porque omite el uso de herramientas y bibliotecas específicas. Además, no hay una capa de abstracción de base de datos, validación de formularios o dependencia de fuentes externas. Es un framework web simple, altamente flexible y de alto rendimiento. Al ser un framework ligero o micro-framework, es fácil de aprender, de usar y de comprender Flask.

Gestor de Base de Datos

MySQL v5.1.38 es la base de datos número uno para Web y es una excelente base de datos embebida. Más de 3.000 Proveedores de Software Independientes (ISVs por sus siglas en inglés) y Fabricantes de Equipos Originales (OEMs por sus siglas en inglés), incluyendo 8 de los 10 mayores y 17 de los 20 principales proveedores de software, de todo el mundo, confían en MySQL como base de datos de sus productos (MySQL. 2013). El sistema de base de datos operacional MySQL es hoy en día uno de los más importantes en lo referente al diseño y programación de bases de datos de tipo relacional. Cuenta con millones de aplicaciones y aparece en el mundo informático como una de las más utilizadas. El programa MySQL se usa como servidor, a través del cual pueden conectarse múltiples usuarios y utilizarlo al mismo tiempo (Definición ABC, 2013).

Es seleccionado MySQL para el trabajo de la base de datos del proyecto ya que es muy destacable su velocidad de respuesta. Se puede utilizar como cliente-servidor o incrustado en aplicaciones. Además,

cuenta con un rico conjunto de tipos de datos y soporta múltiples métodos de almacenamiento de las tablas, con prestaciones y rendimiento diferentes para poder optimizar el SGBD a cada caso concreto.

Conclusiones parciales

- En este capítulo se realizó un análisis del estudio del estado del arte referente a la detección de fallas y la notificación de reportes, lo cual permitió identificar los elementos fundamentales para el desarrollo de la presente investigación.
- El análisis de los sistemas homólogos permitió aportar un grupo de características y funcionalidades que se tomarán como referencia de buenas prácticas en el desarrollo de la solución propuesta.
- Para la propuesta de solución se determina el uso de la metodología de desarrollo AUP-UCI (escenario 4), Visual Paradigm para modelar los diagramas UML, el framework Flask para el desarrollo de la aplicación Web, con el cual es usado el lenguaje de programación Python y Visual Studio Code como editor de código, MySQL como gestor de base de Datos.

CAPÍTULO II: IMPLEMENTACION DEL SISTEMA DIAGNÓSTICO

En el presente capítulo se describen las principales características que debe presentar la solución propuesta, se seleccionará la herramienta necesaria para cumplir los objetivos de la presente investigación. Esta solución es basada en la definición de los requisitos y la arquitectura. Se presentan los principales artefactos propuestos en la fase de modelación en la metodología de desarrollo a aplicar y se fundamentan los patrones de diseño empleados en su elaboración.

2.1 Propuesta de solución

Después de haber analizado los problemas planteados en el capítulo anterior y reflejar que no existe un mecanismo que sea encargado de gestionar el diagnóstico de las fallas de un vehículo, así como el mantenimiento, se ha llegado a la conclusión que es de mucha utilidad para el centro crear una aplicación que se encargue de detectar, almacenar, controlar y predeterminar fallas en un vehículo para la comodidad y seguridad del cliente.

Se propone una aplicación web que permita el control de toda la información referente a las fallas de un vehículo, permitiendo la gestión de dicha información (modificar, insertar, eliminar) ya sea saber el reporte de los vehículos con defecto, el consumo total de combustible y la predicción de alguna falla.

Se dispondrá una sección con los principales resultados del mantenimiento de un carro (informe) y otra sección para la visualización de fallas, notificando al cliente de una posible falla. También tendrá un calendario para dar mantenimiento a los vehículos. Se tendrán 3 niveles de acceso, los cuales serán asignados roles que solo ellos podrán desempeñar y administrar. Uno de los niveles es el Administrador, el cual va a ser el encargado de toda la gestión como tal del sitio.

Mapa conceptual

Los mapas conceptuales son organizadores gráficos que permiten representar el conocimiento como una serie de conceptos que se conectan con palabras vinculadas para formar una proposición, dan una idea clara de conceptos complejos y facilitan la enseñanza-aprendizaje (Vicente García Franco, 2020).

A continuación, el mapa conceptual de la situación problemática.

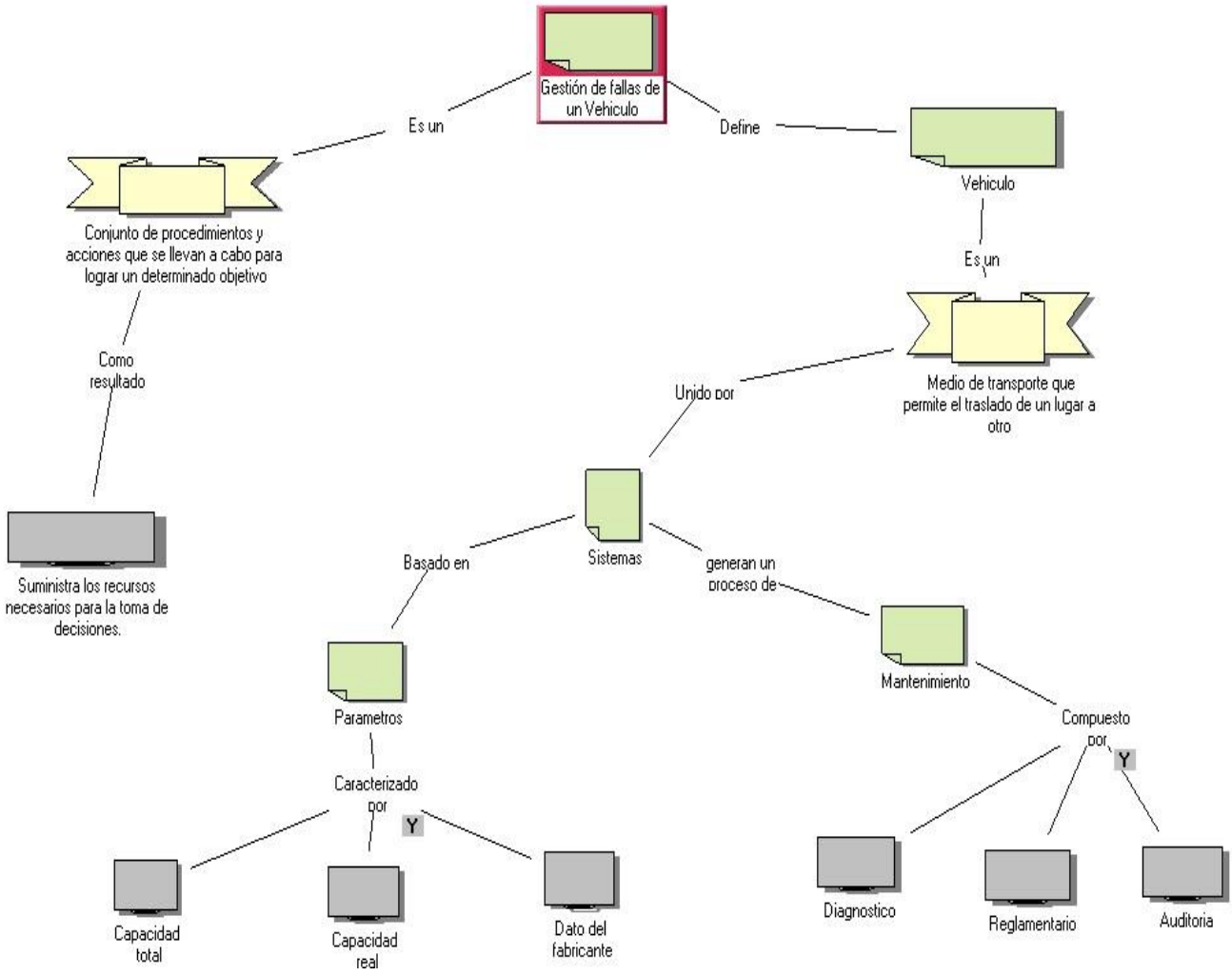

Figura 3: Mapa conceptual. (Fuente: Elaboración Propia).

Definición de las clases

Vehículo: Entidad que posee matricula, marca (que posee un modelo) y está compuesto por sistemas.

Sistema: Entidad que tiene como atributos un nombre, el tipo, un arreglo con los datos de los sensores, parámetros que se van a medir y los tipos de mantenimientos a ejecutar.

Parámetro: Entidad que posee nombre, la magnitud y el valor de lo que se mide.

Mantenimiento: Constituye la parte activa y colaborativa de la gestión de las fallas de un vehículo, presenta un nombre, un tipo y una fecha en la se va a ejecutar.

2.2 Levantamiento de requisitos

Para un sistema los requisitos son descripciones detalladas de las funciones, servicios y las restricciones operacionales de software. El documento de requisitos del sistema define con exactitud lo que se implementará. A menudo, los requisitos de software se clasifican como requisitos funcionales o requisitos no funcionales (Sommerville, 2020).

Para cumplir con la implementación de la propuesta se llevó a cabo formalmente el levantamiento de requisitos. Para ello los requisitos funcionales y no funcionales quedarán descritos detalladamente en el documento, tras haber realizado varias entrevistas al cliente y haber consultado el criterio de expertos respectivamente.

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.

2.2.1 Requisitos funcionales

Los requisitos funcionales para un sistema refieren lo que el sistema debe hacer, ellos dependen del tipo de software que se esté desarrollando, de los usuarios esperados del software y del enfoque general que adopta la organización cuando estos se escriben. Estos varían desde requisitos generales que cubren lo que tiene que hacer el sistema, hasta requisitos muy específicos que reflejan maneras locales de trabajar o los sistemas existentes de una organización (Sommerville, 2020).

A continuación, se muestra un listado de los requisitos funcionales que el sistema deberá cumplir.

Tabla 2: Especificación de requisitos funcionales del sistema. (Fuente: Elaboración Propia).

ID	Nombre	Descripción	Prioridad	Complejidad

RF_1	Agregar vehículo	La aplicación debe agregar un vehículo con los datos adquiridos.	Alta	Media
RF_2	Buscar vehículo	La aplicación debe buscar un vehículo con los datos adquiridos.	Alta	Media
RF_3	Listar vehículo	La aplicación debe mostrar el historial de datos adquiridos.	Alta	Media
RF_4	Visualizar vehículos con defecto	La aplicación debe mostrar el historial de los datos adquiridos.	Alta	Media
RF_5	Actualizar estado de un vehículo	La aplicación debe mostrar los nuevos datos adquiridos.	Alta	Media
RF_6	Visualizar comportamiento de los vehículos	La aplicación debe mostrar el historial de los datos adquiridos.	Alta	Media
RF_7	Obtener distancia total de un vehículo	La aplicación debe almacenar los datos del GPS.	Alta	Media
RF_8	Obtener fecha	La aplicación debe almacenar los datos del GPS.	Alta	Media
RF_9	Obtener kilometraje total de un vehículo	La aplicación debe almacenar los datos del Odómetro.	Alta	Media
RF_10	Visualizar la norma de consumo total	La aplicación debe mostrar una gráfica con el historial de los datos.	Alta	Media

RF_11	Visualizar calendario de mantenimiento	La aplicación debe mostrar un calendario para el mantenimiento de los vehículos.	Media	Alta
RF_12	Mostrar alerta para mantenimiento	La aplicación debe mostrar una alerta para el vehículo que le corresponda el mantenimiento reglamentario.	Alta	Alta

2.2.2 Requisitos no funcionales

Los requisitos no funcionales son limitaciones sobre servicios o funciones que ofrece el sistema, incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Estos se suelen aplicar al sistema como un todo, más que a características o a servicios individuales. Pueden relacionarse con propiedades emergentes, como fiabilidad, tiempo de respuesta y uso de almacenamiento. De forma alternativa, pueden definir restricciones sobre la implementación del sistema, como las capacidades de los dispositivos o las representaciones de datos usados en las interfaces con otros sistemas. Los requisitos no funcionales, como el rendimiento, la seguridad o la disponibilidad, especifican o restringen por lo general características del sistema como un todo (Pressman 2021).

A continuación, se exponen los requisitos no funcionales de la propuesta de solución:

Tabla 3: Especificación de requisitos no funcionales del sistema. (Fuente: Elaboración Propia).

ID	Característica	Descripción	Atributo
RNF_1	Usabilidad	El sistema debe ser fácil de utilizar para los usuarios que tengan niveles básicos de computación o hayan trabajado con la Web (.	Capacidad de Aprendizaje
RNF_2		Debe tener una opción de ayuda sobre las principales funcionalidades que brinda el sistema y sus iconos respectivos, para un mejor entendimiento.	Capacidad para reconocer su adecuación
RNF_3	Eficiencia de desempeño	La aplicación debe estar concebida para el consumo mínimo de recursos.	Rendimiento y Resistencia

RNF_4	Compatibilidad	Capacidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.	Interoperabilidad.
RNF_5	Seguridad	Capacidad de protección contra el acceso de datos e información no autorizados, ya sea accidental o deliberadamente.	Confidencialidad Autenticidad
RNF_6	Mantenibilidad	Facilidad con la que se pueden establecer criterios de prueba para el sistema y con la que se pueden llevar a cabo las pruebas para determinar si se cumplen dichos criterios.	Capacidad para ser Probado
RNF_7	Portabilidad	Capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hardware, software, operacionales o de uso.	Adaptabilidad.

2.2.3 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 de implementarlas sin dificultad (Sanchéz, 2015).

A continuación, se muestra una de las historias de usuario, las demás historias de usuario están en el Anexo 2.

Tabla 4: Historia de Usuario Registrar Vehículo. (Fuente: Elaboración Propia).

Historia de usuario	
Número: 1	**Número del requisito:** Añadir Vehículo
Programador: Amanda Ojeda Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora

Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 1 hora
Descripción: **Objetivo:** Añadir un vehículo con sus atributos a la base de datos del sistema y mostrarlo en la lista de vehículos de la aplicación. **Precondiciones**: Se debe estar registrado en la página. **Acciones a realizar:** 1. Presionar el botón Añadir. 2. Rellenar los campos con las características del vehículo (matricula, marca, modelo, ancho, largo, peso). 3. Presionar el botón de Aceptar.	
Prototipo de interfaz:	

2.2.4 Diagrama de clase del diseño

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, en la Figura 4 se muestra el DCD del requisito funcional Gestionar Modificación del Estado del vehículo. Los DCD correspondientes a los requisitos funcionales restantes se muestran en el Anexo 3.

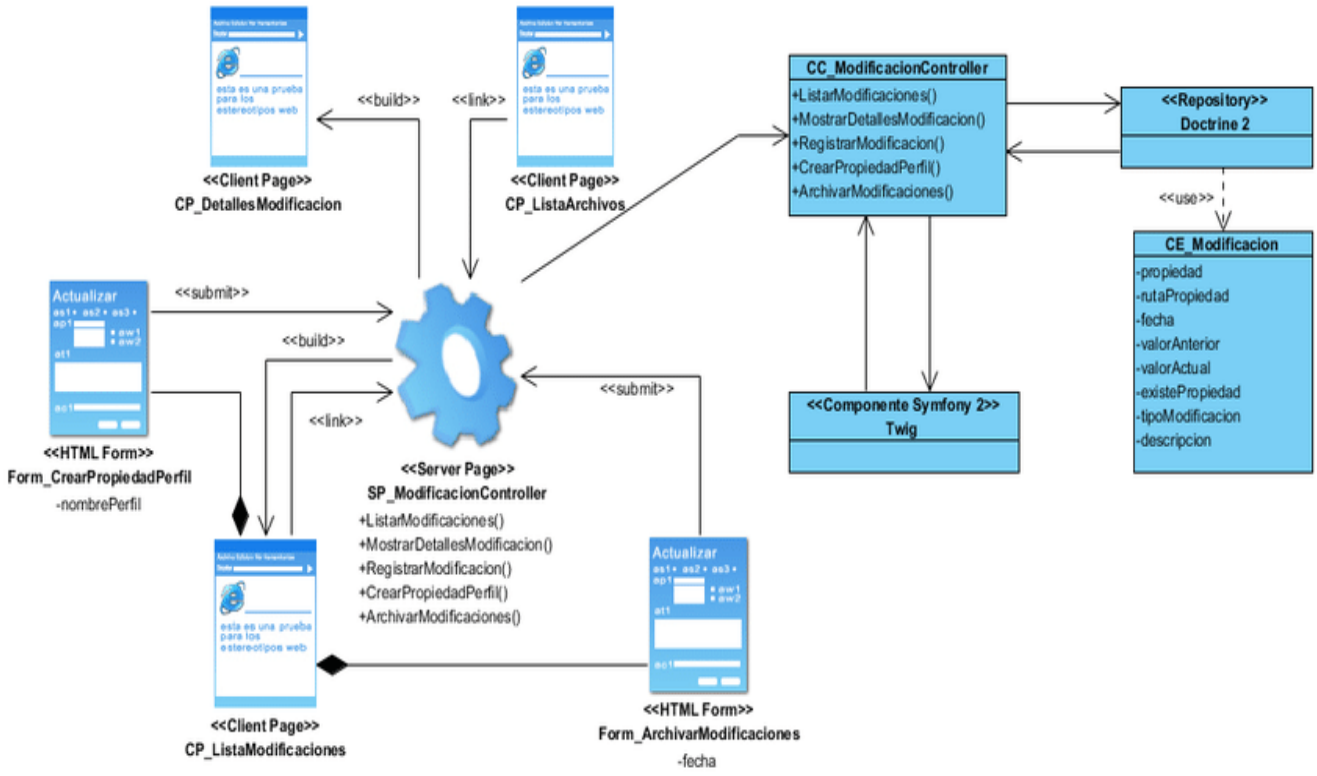

Figura 4: Diagrama de clase de Diseño (Fuente: Elaboración propia).

2.3 Arquitectura del sistema

La arquitectura de software conforma el esqueleto de cualquier sistema, y es la principal responsable de los atributos de calidad del sistema (parte de los requerimientos no funcionales del mismo) la misma identifica los elementos más importantes de un sistema, así como sus relaciones, es decir, da una visión global del sistema (Pressman, 2021).

Patrón Modelo Vista Plantilla

Es un patrón de arquitectura de software utilizado por el framework del lenguaje de programación Python, Flask. Es una variante al MVC. (Universidad de Alicante, 2020).

Este patrón arquitectónico está compuesto por tres componentes:

Modelo: Es la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.

Vista: Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: puedes pensar en esto como un puente entre el modelo y las plantillas.

Plantilla: Es la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: como algunas cosas son mostradas sobre una página web u otro tipo de documento.
En los siguientes puntos se detalla el funcionamiento del MCT:

- El navegador envía una solicitud.
- La vista interactúa con el modelo para obtener datos.
- La vista llama a la plantilla.
- La plantilla renderiza la respuesta a la solicitud del navegador.

2.3.1 Patrones de diseño

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular. Tienden a ser independientes de los lenguajes y paradigmas de programación y su aplicación no afecta necesariamente al sistema completo, pero si a un subsistema o parte del mismo (Pressman, 2010).
A continuación, se muestran los patrones de diseño empleados en la implementación de la propuesta:

Experto: Se encarga de asignar la responsabilidad al experto en la información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Brinda la posibilidad de conservar el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide, lo que provee un bajo nivel de acoplamiento. Promueve clases sencillas y cohesivas que son más fáciles de mantener y comprender (Pressman, 2010). Se pone en práctica en las clases Vehículo que es la que posee toda la información necesaria para mostrar y almacenar los datos adquiridos.

Figura 5 clase Routes (Fuente: Elaboración propia)

Creador: Consiste en asignar a un Objeto la responsabilidad de crear otro Objeto. Un objeto es responsable de crear una nueva instancia de alguna clase si: agrega o contiene objetos de ella, registra las instancias de sus objetos o tiene los datos de inicialización que serán enviados a ella cuando el objeto sea creado (Pressman, 2010). 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.
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). En la propuesta de solución se utilizó este patrón en clases como *Account* la cual se relacionan con la menor cantidad de clases posibles para el cumplimiento de sus funciones.

Controlador: El patrón ofrece una guía para tomar decisiones sobre los eventos de entrada, asignando la responsabilidad del manejo de mensajes de los eventos del sistema a una clase controladora, ya que los elementos de interfaz y sus controladores de eventos, no deben ser responsables de controlar los eventos del sistema (Pressman,2010). 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 *Users*, la cual va a controlar las funciones sobre la interfaz principal.

```
class Users(db.Model, UserMixin):

    __tablename__ = 'Users'

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), unique=True)
    email = db.Column(db.String(64), unique=True)
    password = db.Column(db.LargeBinary)

    def __init__(self, **kwargs):
        for property, value in kwargs.items():
```

Figura 5: Patrón Controlador usado en la propuesta de solución. (Fuente: Elaboración propia)

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

Patrones Gang of Four (GOF)

Los patrones de diseño son soluciones a los problemas de diseño de software que se encuentran una y otra vez en el desarrollo de aplicaciones del mundo real. Los patrones tratan sobre diseños reutilizables e interacciones de objetos. Los 23 patrones de *Gang of Four* (GoF) generalmente se consideran la base de todos los demás patrones. Los patrones GoF son soluciones concretas y técnicas basadas en la Programación Orientada a Objetos (POO) (Dofactory, 2020). Estos patrones se dividen en tres categorías: los creacionales, los estructurales y los de comportamiento, a continuación, se muestran los que se implementaron en el sistema:

Patrones de creación

• *Singleton*: garantiza que una clase solo tenga una instancia y proporcionar un punto de acceso global a ella. En la propuesta de solución se utilizó este patrón en la instancia de las clases, ejemplo de ello la instancia a las clases Vehículo, dando un punto de acceso global a la clase.

```
    # obtener listado de todos los vehiculos #
    sql_todos_vehiculos = 'select * from vehiculo;'
    mydb.cursor.execute(sql_todos_vehiculos)
    # 'data' almacena los datos recuparados de la base de datos
    datos = mydb.cursor.fetchall()

    # obtener listado de los vehiculos con fallas
    sql_fallas_vehiculos = '''select vehiculo.matricula as Matricula, vehiculo.marca as Marca, falla.descripcio
                        join vehiculo on vehiculo.id = falla_vehiculo.id_vehiculo;'''
    mydb.cursor.execute(sql_fallas_vehiculos)
    datos_fallas = mydb.cursor.fetchall()
```

Figura 6: Patrón de creación usado en la propuesta de solución. Instancia a la clase Vehículo. (Fuente: Elaboración propia)

• Fachada: brinda una interfaz que abstrae completamente al usuario de la complejidad de los procesos. En la propuesta de solución se utilizó este patrón en las interfaces del proyecto ubicadas en el directorio./templates/, donde se encuentran las acciones definidas para las operaciones.

2.4 Modelo de Base de Datos

Un modelo de datos es un lenguaje orientado a describir una Base de Datos. Típicamente un modelo de datos permite describir las estructuras de datos que representan a los elementos del dominio, el tipo de dato de los atributos y la forma en que se relacionan (Pressman, 2010).

A continuación, se muestra el modelo de datos de la solución:

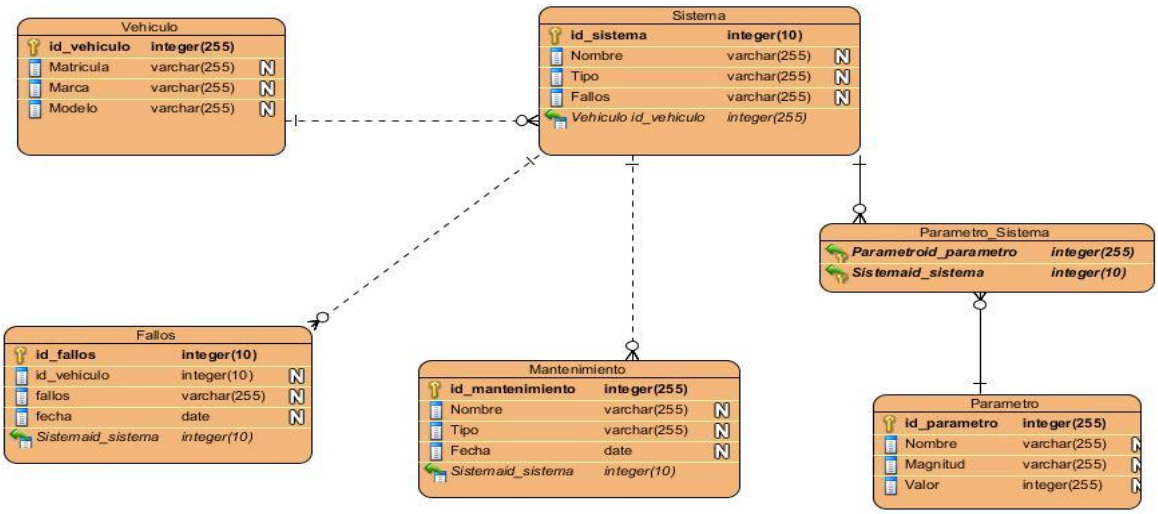

Figura 7:Diseño de base de datos. Diagrama Entidad-Relación. (Fuente: Elaboración propia).

Conclusiones del Capitulo

El análisis y diseño de la propuesta permitió realizar una descripción detallada de las características del Sistemas, lo que permitió un mejor entendimiento para la fase de implementación al tener los principales artefactos para el desarrollo. Se definieron los requisitos funcionales y no funcionales, los cuales proporcionan una guía de desarrollo de las funcionalidades del sistema.

1. La utilización del patrón arquitectónico modelo-vista-controlador (MVC) permitió estructurar los componentes del módulo, de esta manera, se obtuvo una clara separación entre interfaz, lógica de negocio y datos, lo cual sirvió de soporte a la reutilización de los componentes reduciendo el costo y tiempo de desarrollo.
2. El modelo de datos permite describir la estructura de la base de datos con el fin de garantizar una correcta gestión de los mismos.
3. Los diagramas de clases del diseño permitieron establecer la base estructural de la implementación de la propuesta de solución.
4. La selección los patrones GRASP y GoF en la implementación permitió una adecuada reutilización del código para futuras versiones.

CAPÍTULO III: VALIDACIÓN DEL SISTEMA DIAGNÓSTICO

Introducción

En el presente capitulo se evalúa el grado de calidad y fiabilidad de los resultados obtenidos en el desarrollo de la presente investigación. Se muestran los resultados en cada caso luego de aplicar las técnicas de validación de los requisitos tanto al diseño del sistema como a la solución desarrollada. Para la validación del diseño se aplican métricas y para la validación de la solución se define una estrategia de prueba aplicando los 4 niveles de pruebas con sus respectivos métodos y técnicas. Por otra parte, se realiza una verificación del cumplimiento del objetivo general de la investigación.

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.

3.1.1 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.

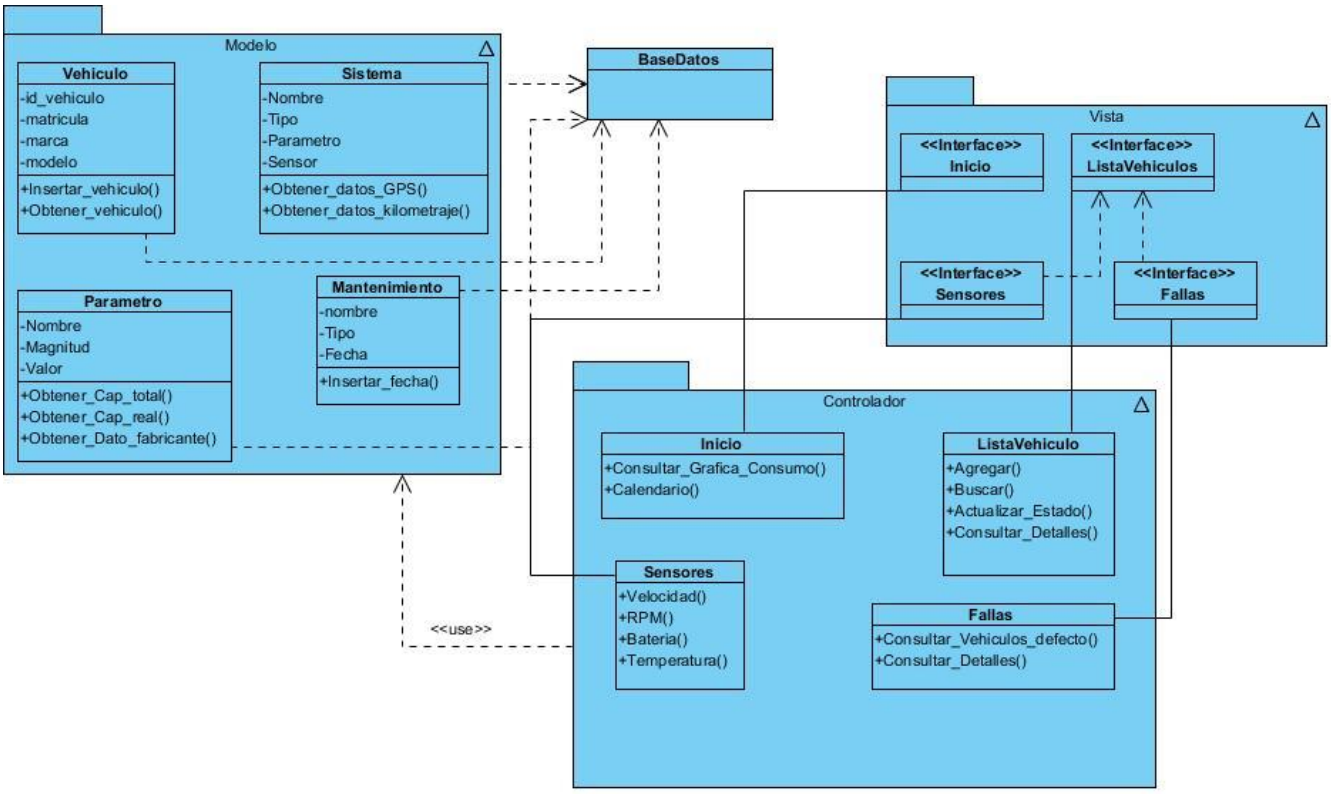

Figura 8: Diagrama de componentes del sistema (Fuente: Elaboración propia).

3.1.2 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.

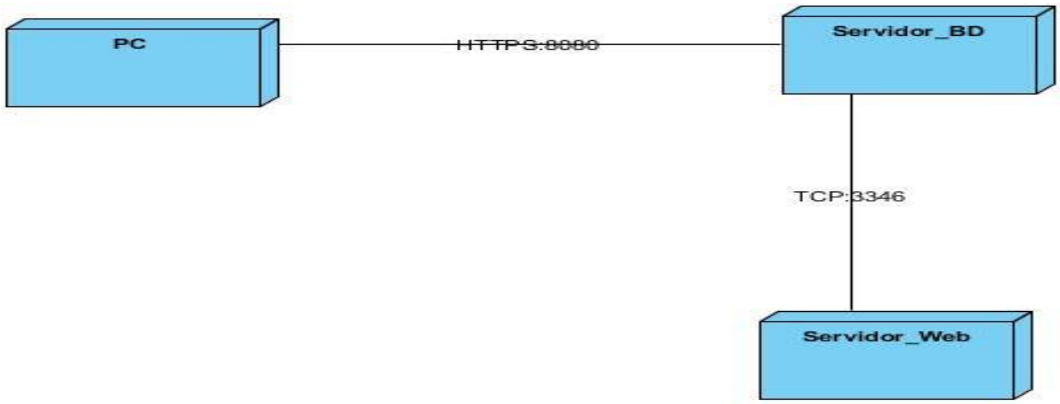

Figura 9: Diagrama de Despliegue (Fuente: Elaboración propia).

3.2 Validación de requisitos

Con el objetivo de comprobar que los requisitos del software obtenido se ajustan al componente que se necesita, se realizó un proceso de validación de los mismos, para el cual se emplearon las siguientes técnicas:

Revisiones de los requisitos: Se realizaron revisiones a cada uno de los requisitos por parte del equipo de desarrollo. Las revisiones internas generaron un total de 3 no conformidades (NC), las cuales fueron corregidas satisfactoriamente. Con el equipo de análisis y líder de proyecto, se realizó la revisión en la que se aprobaron finalmente los mismos.

Generación de casos de prueba: Como parte del proceso de validación de los requisitos funcionales se diseñaron casos de pruebas, se definieron 12 diseños de casos de prueba, uno por cada requisito, teniendo como referencia el diseño de caso de prueba del RF2 Visualizar vehículos con defecto.

Diseño de prototipos web: Se mostró al cliente un modelo ejecutable que permitió tener una visión preliminar de cómo se vería el componente y a través de la interacción con estos se comprobó la aprobación y satisfacción del cliente hacia los prototipos diseñados.

3.3 Métricas aplicadas a los requisitos

Con el fin de medir la calidad de la especificación de los requisitos se aplicó la métrica Calidad de la especificación (CE), una de las métricas propuestas para la metodología AUP (UCI). A continuación,

se muestra el cálculo del total de requisitos de la especificación con el objetivo de obtener qué entendibles y precisos son los mismos:

- **Nr**: total de requisitos de especificación.
- **Nf**: cantidad de requisitos funcionales.
- **Nnf:** cantidad de requisitos no funcionales.

Teniendo en cuenta que: Nr = Nf + Nnf

Nr = 12 + 7

Nr = 19

Como resultado de la sustitución de los valores para el componente se obtiene que: el total de requisitos de especificación tiene un valor de 19, quiere decir que, existe un total de 19 requisitos para ser analizados y tras este se procede a determinar cuán entendibles y precisos son.

Especificidad de los Requisitos (ER)

- **Nui:** número de requisitos para los cuales todos los revisores tuvieron interpretaciones idénticas.

Mientras más cerca de 1 esté el valor de ER, menor será la ambigüedad. Para determinar, la Especificidad de los Requisitos (ER) o ausencia de ambigüedad en los mismos se realiza la siguiente operación:

Teniendo en cuenta que: ER = Nui / Nr

Para el caso de los requisitos obtenidos del componente, tres produjeron contradicción en las interpretaciones. Sustituyendo las variables se obtiene:

ER = 16/ 19

ER = 0.84

Obteniéndose así, un resultado final satisfactorio, con un grado de ambigüedad de los requisitos bajo (19%) ya que el 84% son entendibles. Los requisitos ambiguos fueron modificados y validados para garantizar una correcta interpretación.

3.4 Validación del diseño

La validación del diseño a través de las métricas, permite medir de forma cuantitativa la calidad de los atributos internos del software, de forma tal que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente (Pressman, 2010).

Con el objetivo de comprobar la calidad del diseño se emplearon las métricas de diseño relaciones entre clases (RC) y tamaño operacional de clase (TOC).

Relaciones entre clases (RC)

La métrica RC está dada por el número de relaciones de uso de una clase con otra. Lo primero es evaluar un conjunto de atributos de calidad entre los que se encuentran el acoplamiento, complejidad de mantenimiento y reutilización de cada clase (Pressman, 2010).

A continuación, se exponen las actividades que se llevaron a cabo para aplicar la métrica a todas las clases del componente propuesto:

- Determinar la Cantidad de Relaciones de Uso (CRU) que poseen las clases a medir.
- Calcular el promedio de las CRU.

Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos por Lorenz, 1994(Lorenz, y otros, 1994)**.**

En la Figura 10 se muestra el resultado de aplicar la métrica RC:

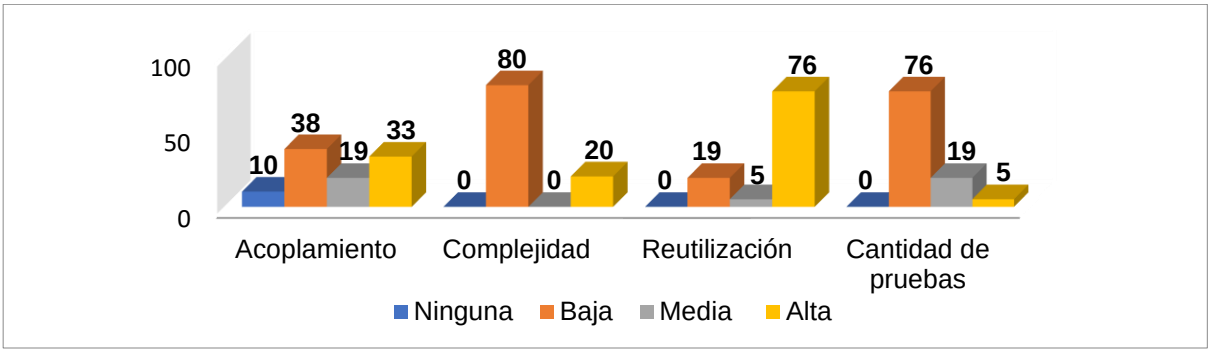

Figura 10: Representación en (%) de los resultados de la aplicación de la técnica RC (Fuente: Elaboración propia).

Acoplamiento: según los resultados que se muestran, el 10% de las clases no posee relaciones de uso y el 38 % posee un acoplamiento bajo, el 19% un acoplamiento medio y el 33 % un acoplamiento alto, por lo que la mayoría de las clases poseen valores de acoplamiento nulo, bajos y medio, validando una realización correcta del diseño.

Complejidad de mantenimiento: según los resultados que se muestran en la figura anterior, el 80% de las clases presentan una complejidad de mantenimiento baja, demostrando que son de fácil soporte.

Reutilización: según los resultados que se muestran en la figura, el 76% de las clases tiene un grado alto de reutilización.

Cantidad de pruebas: luego de aplicar la métrica se obtuvo que el 76% de las clases poseen un grado bajo de esfuerzo a la hora de realizar cambios, rectificaciones y pruebas de software.

Teniendo en cuenta lo anterior se concluye que el diseño del sistema propuesto en la presente investigación alcanzó resultados positivos durante la evaluación de la métrica. Se obtuvo un 38% de bajo acoplamiento, 80% de baja complejidad de mantenimiento, el 76% presenta un bajo grado de esfuerzo destinado a pruebas y una reutilización alta.

Tamaño Operacional de Clases (TOC)

La métrica TOC fue aplicada a cada una de las clases del diseño con el objetivo de medir la calidad de las mismas con respecto a su grado de responsabilidad, complejidad de implementación y reutilización (Pressman, 2010).

A continuación, se explican los pasos que se llevaron a cabo para aplicar la métrica:

- Cálculo del umbral. El umbral se toma del tamaño general de una clase que se determina sumando todas las operaciones que posee.
- Calcular el promedio de los umbrales.
- Teniendo en cuenta los valores antes obtenidos se determina la incidencia de los atributos de calidad en cada una de las clases, según los criterios expuestos por Lorenz, 1994.

En la Figura 11 se muestra el resultado de aplicar la métrica TOC:

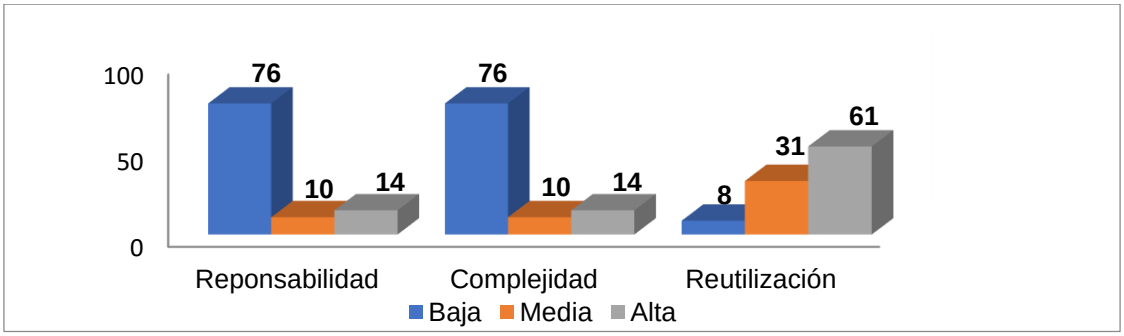

Figura 11: Representación en (%) de los resultados de la aplicación de la técnica TOC

(Fuente: Elaboración propia)

Responsabilidad: después de aplicar la métrica, se obtuvieron resultados satisfactorios que reflejan una responsabilidad baja con un valor del 76%.

Complejidad de implementación: luego de haber realizado la medición de la métrica, se obtuvieron resultados positivos ya que la complejidad de las clases es baja en un 76%.

Reutilización: se obtuvieron valores que según muestra la gráfica de la figura anterior la reutilización se comporta en un nivel alto con un 61%.

Teniendo en cuenta lo anterior se concluye que el diseño del sistema propuesto en la presente investigación alcanzó resultados positivos durante la evaluación de la métrica. Se obtuvo un 76% de baja responsabilidad de las clases y complejidad de implementación respectivamente, mientras que la reutilización es alta para un 61%. Estos datos muestran que las clases tienen poca responsabilidad, lo cual amplía su reutilización y facilita la implementación.

3.5 Pruebas de software

Las pruebas de software comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo (PMOinformatica.com, 2020). La metodología seleccionada para guiar el proceso de desarrollo de software define las pruebas como una de las disciplinas a tener en cuenta.

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, 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 = 12/12

C = 1

- Pruebas unitarias

Las pruebas unitarias son una forma de comprobar que un fragmento de código funciona correctamente, consisten en aislar una parte del código y comprobar que funciona a la perfección;

pequeñas pruebas que validan el comportamiento de un objeto y la lógica (Yeeply, 2019). Para aplicar las pruebas unitarias, se utiliza el método de Caja blanca. Con el empleo de este método es posible desarrollar casos de prueba que garanticen la ejecución, al menos una vez, de los caminos independientes (Pressman, 2010). Para aplicar este método se define la técnica de ruta básica.

Técnica de ruta básica

La técnica de ruta básica es empleada en el método de Caja blanca, la misma tiene como objetivo comprobar que cada camino se ejecute independiente de un componente o programa, obteniéndose una medida de la complejidad lógica del diseño. Esta técnica de prueba debe ser utilizada para evaluar la efectividad de los métodos asociados a una clase, con el objetivo de asegurar que cada camino independiente sea ejecutado por lo menos una vez en el sistema (Pressman, 2010).

Pressman propone como estrategia para aplicar la ruta básica, realizar un análisis de la complejidad ciclomática de cada procedimiento que componen las clases del sistema; una vez concluido este paso se selecciona el método con valor de contener errores, además ofrece una medida del número de pruebas que deben diseñarse para validar la correcta implementación de una función determinada.

Se derivan casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control, para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática. A continuación, se muestra un ejemplo de la aplicación de la técnica a uno de los métodos más complejos desde el punto de vista de implementación (vehiculo_matricula_buscar) perteneciente a la clase app:

```
247  @app.route('/reporte/<string:username>/<string:session>/buscar', methods=['GET', 'POST'])
248  def buscar():
249      if username in logged_in and (logged_in[username]['object'].session_id == session):
250          user = {
251              "username" : username,
252              "image":"/static/images/amanSingh.jpg",
253              "api": logged_in[username]["object"].api,
254              "session" : session,
255
256          }
257          print(request.method)
258      if request.method == 'POST':
259          id_matricula = request.form['vehiculo_matricula_buscar']
260
261          mydb.cursor.execute ("""
262          query = "SELECT * FROM vehiculo WHERE matricula=%s;"
263          mydb_aux.cursor.execute(query)
264          """, (id_matricula))
265          data = mydb.fetchall()
266          print("Data Updated Successfully")
267          return redirect(url_for('reporte', username=username, session=session,  data=data))
268
269      print("Data Updated UnSuccessfully")
270      return redirect(url_for('reporte', username=username, session=session ))
```

1

2

3

4

Figura 12: Método vehiculo_matricula_buscar de la clase app (Fuente: Elaboración propia).

A continuación, se detallan los pasos que se utilizaron al aplicar la técnica ruta básica:

1.Confeccionar el grafo de flujo: usando el código de la Figura 17 se realizó la representación del grafo de flujo, el cual describe un flujo de control lógico y está compuesto por los siguientes elementos:

- Nodos de gráfica de flujo: son círculos que representan una o más instrucciones procedimentales.
- Aristas o enlaces: son flechas que representan el flujo de control y son análogas a las flechas del diagrama de flujo.
- Regiones: son las áreas delimitadas por aristas y nodos.

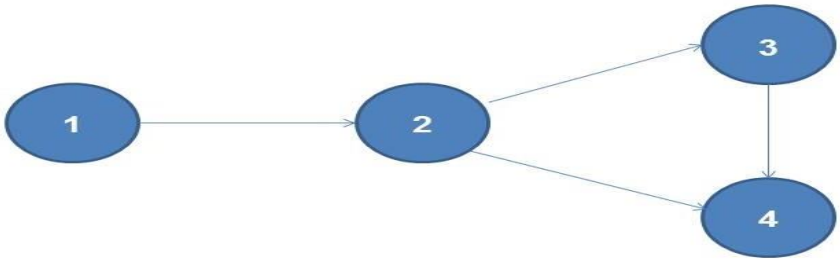

Figura 13:Grafo de la ruta básica del método vehiculo_matricula_buscar (Fuente: Elaboración propia).

2. Calcular la complejidad ciclomática: brinda una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse para asegurar que todas las instrucciones se hayan ejecutado al menos una vez. Se calcula mediante las tres formas siguientes:

- El número de regiones del gráfico de flujo (Pressman, 2010). El gráfico de flujo tiene dos regiones.
- V (G) = E – N + 2, donde E es el número de aristas del grafo de flujo y N es el número de nodos del mismo (Pressman, 2010).

V (G) = E – N + 2 = 4 aristas – 4 nodos + 2 = 2

- V (G) = P + 1, donde P es el número de nodos predicados contenidos en el gráfico de flujo (Pressman, 2010).

V (G) = 1 nodo predicado + 1 = 2

3. Determinar un conjunto básico de rutas linealmente independientes: el valor de V (G) indica el número de rutas linealmente independientes de la estructura de control del programa, por lo que se definen los 2 caminos obtenidos:

Ruta Básica 1: 1-2-3-4

Ruta Básica 2: 1-2-4

Cada nodo que contiene una condición y está caracterizado porque dos o más aristas emergen de él.

3. Obtención de casos de pruebas: cada ruta independiente es un caso de prueba a realizar, de forma que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino. En este caso se obtuvieron 2 caminos básicos, que dan lugar a la confección de igual número de casos de pruebas, para aplicarlas a este método. A continuación, se muestran los casos de pruebas:

Tabla 5: Caso de prueba de caja blanca para el camino básico #1. Fuente (Elaboración Propia).

Caso de prueba Camino Básico #1	
Descripción: Este camino permite que el método vehiculo_matricula_buscar se ejecute cuando el campo Vehículo no presenta valores nulos.	
Entrada	Una matrícula.
Resultado	Muestra los reportes en tiempo real que se encuentran en el vehículo con la matrícula especificada.
Condiciones	Deben estar especificadas ambas fechas y la fecha de inicio debe ser menor igual que la fecha final.

Tabla 6: Caso de prueba de caja blanca para el camino básico #2. (Fuente: Elaboración propia)

Caso de prueba Camino Básico #1	
Descripción: este camino permite que el método vehiculo_matricula_buscar se ejecute cuando el campo Vehículo presenta valores nulos.	
Entrada	Ninguna
Resultado	No muestra ningún dato del sistema.
Condiciones	El campo Vehículo debe estar vacío.

Descripción de la ejecución de los casos de prueba

Para ejecutar cada caso de prueba se realizaron pruebas manuales al código probando cada camino descrito a través de los casos. En el caso de prueba # 1 se introdujo una matrícula, mostrándose en el reporte los datos para el vehículo de la matrícula especificada, esto evidencia que el caso de prueba se ejecutó satisfactoriamente. En el caso de prueba # 2 no se introduce el dato en el campo Vehículo y no muestra ningún dato del sistema, esto evidencia que el caso de prueba # 2 también se ejecutó satisfactoriamente.

Una vez ejecutados los dos casos de pruebas obtenidos a través de la aplicación de la técnica ruta básica, se concluye que los mismos fueron probados satisfactoriamente, demostrando que todas las rutas de este código se ejecutaron al menos una vez.

3.6 Método de caja negra

Las pruebas de caja negra, también denominadas, pruebas de comportamiento, se concentran en los requisitos funcionales del software. Es decir, permiten al ingeniero de software derivar conjuntos de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. La prueba de caja negra no es una opción frente a las técnicas de caja blanca. Es, en cambio, un enfoque complementario que tiene probabilidades de describir una clase diferente de errores de los que se descubrirán con los métodos de caja blanca (Pressman, 2010).

Las pruebas de caja negra tratan de encontrar errores en las siguientes categorías:

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y término (Pressman, 2010).

Para desarrollar el método de caja negra existen varias técnicas, entre ellas están (Pressman, 2010):

Técnica de la partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Técnica del análisis de valores límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Con el objetivo de comprobar el funcionamiento del sistema se realizaron un total de tres iteraciones de pruebas, para poder alcanzar resultados satisfactorios, atendiendo al correcto comportamiento del sistema ante diferentes situaciones. Se realizaron las pruebas a nivel de integración y sistema,

aplicando el método de caja negra con el uso de las técnicas partición de equivalencia y análisis de valores límites. A continuación, en la figura 14 se muestran los resultados de la aplicación del método, ejecutado por especialistas del equipo de calidad de Xetid.

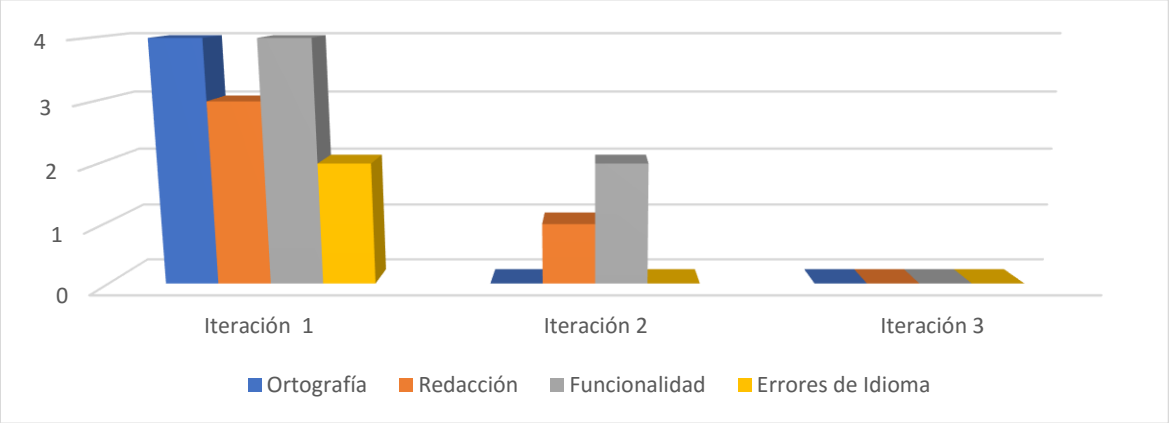

Figura 14: No conformidades detectadas al aplicar el método de caja negra (Fuente: Elaboración propia).

Como muestra la figura 22, en la primera iteración se detectaron un total de 13 No Conformidades (NC), clasificadas en 4 de ortografía, 3 de redacción, 4 de funcionalidad y 2 de errores de idioma. Luego en una segunda iteración persistieron 1 de redacción y 2 de funcionalidad. En la tercera iteración los resultados fueron satisfactorios, obteniéndose cero NC.

3.7 Pruebas de Compatibilidad

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.

Pruebas de Integración

Las pruebas de integración se realizan para probar que todos los elementos (módulos) que componen el sistema funcionen correctamente las relaciones entre ellos.

Tabla 7: pruebas de integración. (Fuente: Elaboración Propia).

Sistema	**Funcionalidades**	**Resultado de la prueba**
Sistema de Gestión en flotas de vehículos para la empresa Xetid.	Instalación y configuración del sistema Diagnóstico de la empresa Xetid.	El sistema se integra satisfactoriamente logrando los procesos de gestión de la flota de vehículos.

Pruebas de rendimiento

La prueba del rendimiento ocurre a lo largo de todos los pasos del proceso de prueba. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme se realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema (Pressman, 2021).

Las pruebas de rendimiento por lo general requieren instrumentación de *hardware* y de *software*, es decir, con frecuencia es necesario medir la utilización de los recursos (por ejemplo, ciclos del procesador) en forma meticulosa. La instrumentación externa puede monitorear intervalos de ejecución y eventos de registro (por ejemplo, interrupciones) conforme ocurren, y los muestreos del estado de la máquina de manera regular. Con la instrumentación de un sistema, la persona que realiza la prueba puede descubrir situaciones que conduzcan a degradación y posibles fallas del sistema (Pressman, 2021).

Pruebas de carga

Mediante la ejecución de las pruebas de Carga es posible identificar la capacidad de recuperación de un sistema cuando es sometido a cargas variables tanto de usuarios como de procesos. Al realizar las pruebas de carga se puede determinar el tiempo de respuesta de todas las transacciones críticas del sistema y encontrar cuellos de botella de la misma (Pressman, 2021).

Pruebas de estrés

Mediante las pruebas de estrés es posible identificar la capacidad de respuesta de un sistema bajo condiciones de carga extrema, representadas por una alta concurrencia de Usuarios y/o procesos, una vez realizadas las pruebas de estrés se podrá conocer el punto de quiebre del aplicativo en términos de capacidad de respuesta, con lo cual será posible establecer acciones de optimización en diferentes niveles para asegurar una mejor capacidad de concurrencia de usuarios y/o procesos que se verá reflejada en una óptima operación de negocio (Pressman, 2021).

Resultados de las pruebas de rendimiento

Para la realización de las pruebas de carga y estrés al sistema Diagnóstico se utilizó el software Apache JMeter en su versión 2.8.4.

Hardware de prueba (PC cliente):

- Tipo de procesador: Intel(R) Core(TM) 2 Duo CPU @ 2.26GHz 2.27GHz.
- RAM: 3 GB DDR2.
- Tipo de Red: Ethernet 10/100Mbps.

Hardware de prueba (PC servidor):

- Tipo de procesador: Intel(R) Core(TM) i3-6100u @ 2.30GHz 2.30GHz.
- RAM: 6 GB DDR3.
- Tipo de Red: Ethernet 10/100Mbps.

Software instalado en ambas PC:

- Tipo de servidor web: Apache.
- Plataforma: Windows 10 Home (PC cliente y servidor)
- Servidor de BD: MySQL v5.1.38.

La prueba define doscientos hilos de concurrencia, los cuales simulan doscientos usuarios accediendo concurrentemente. Se simularon un total de mil peticiones a cinco direcciones del módulo en el servidor. En la Tabla 8 se puede observar los resultados obtenidos por el sistema.

Tabla 8: Resultados de las pruebas de rendimiento. (Fuente: Elaboración Propia).

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/se
home	200	2281	2172	3527	3909	4450	304	4694	72,50%	42,1/sec	109,69	7,5
ficha tecnica	200	4396	2005	10513	10863	11998	1742	12102	64,00%	12,2/sec	292,55	0,8
insertar auto	200	6431	8872	11153	11435	11744	2000	11798	49,00%	8,1/sec	218,15	0,8
insertar m...	200	4803	8236	8309	9032	9273	0	9295	100,00%	7,0/sec	13,65	0,0
dti	200	4677	2023	10057	10370	10964	892	11044	46,50%	9,6/sec	286,41	1,0
Total	1000	4518	2141	10499	10757	11585	0	12102	66,40%	28,4/sec	483,01	2,6

Descripción de los parámetros evaluados:

Muestras: cantidad de hilos utilizados para la URL.

Media: tiempo promedio en milisegundos para un conjunto de resultados.

Min: tiempo mínimo que demora un hilo en acceder a una página.

Max: tiempo máximo que demora un hilo en acceder a una página.

%Error: porciento de error de las respuestas de las peticiones.

Rendimiento: rendimiento medido en los requerimientos por segundo / minuto / hora.

Kb/s Recibidos: rendimiento medido en Kbyte por segundo.

3.8 Pruebas de Seguridad

Las pruebas de seguridad buscan medir la Confidencialidad, Integridad y Disponibilidad de los datos, desde la perspectiva del aplicativo, es decir partiendo a identificar amenazas y riesgos desde el uso o interface de usuario final. Una vez ejecutadas las pruebas de seguridad es posible medir y cuantificar los riesgos a los cuales se ven expuestos los aplicativos tanto en la infraestructura interna como externa, valiéndose de la filosofía del Hacking ético. Al módulo desarrollado se le realizaron una serie de pruebas de seguridad mediante el software Acunetix, las cuales se presentan a continuación:

- Ataques de inyección SQL.
- Cross-Site Scripting (XSS).
- Falsificación de petición (CSRF).

Vulnerabilidades detectadas

Tabla 9: Pruebas de seguridad Tipo Cantidad Descripción Fuente (Elaboración Propia).

Tipo	**Cantidad**	**Descripción**
Implementación	20	Formularios HTML sin protección CSRF.
Configuración	5	La configuración del servidor de aplicaciones permite la visualización de información sensible del negocio.

Observaciones:

La prueba realizada mediante la herramienta Acunetix devolvió que se detectan veinticinco no conformidades relacionadas con la configuración del servidor de aplicaciones y la implementación del software, solucionando las vulnerabilidades detectadas que pudieran poner en riesgo la seguridad e integridad del módulo.

3.9 Validación de los resultados de investigación

Las deficiencias descritas en la introducción de la presente investigación afectan directamente el cumplimiento de actividades fundamentales en el diagnóstico de vehículos. Para demostrar la existencia del problema y aplicar los instrumentos seleccionados se definieron las siguientes variables:
Variable dependiente: mantenimiento del vehículo.
Variable independiente: gestionar diagnóstico.
Una vez obtenidos los resultados de la encuesta realizada a los especialistas de la Xetid y a partir de las pruebas realizadas correspondientes al sistema de la presente investigación se evidencia una reducción considerable del tiempo en la gestión de la información. Quedando en evidencia cómo, con el desarrollo del sistema para la gestión de diagnóstico para vehículo, se contribuirá al mantenimiento y control de vehículos.
Una vez realizada la evaluación de las variables, analizados los resultados y las características de la solución de la presente investigación se demuestra la validez de la idea a defender al plantear que, con el desarrollo de un sistema para el diagnóstico de vehículos mediante el dispositivo OBD2, se contribuirá a la control y correcto mantenimiento en el proceso.

Conclusiones del capítulo

En el presente capítulo se logró ratificar que los requisitos de software definen el sistema que el cliente desea al aplicar las técnicas de validación definidas para los mismos. Se elaboró el diagrama de componentes lo que permitió visualizar con facilidad la estructura general de la solución. Además, el diagrama de despliegue sirvió como guía para el correcto despliegue del sistema. De igual manera, se realizó la validación del diseño mediante las métricas TOC y RC, lo que permitió definir de forma cuantitativa la calidad del mismo en el desarrollo del componente para la gestión del diagnóstico de fallas de los vehículos. Por otra parte, la ejecución de pruebas funcionales mediante el método caja negra demostró que las funciones son operativas a través de la interfaz del software, manteniendo así la integridad de la información externa. La ejecución de pruebas unitarias mediante el método caja blanca demostró que las funciones internas son operativas, facilitando la detección de no conformidades para

su corrección. Por último, el análisis del comportamiento de la variable que forma parte del problema de la investigación, demostró que el componente desarrollado contribuye a la celeridad en el proceso.

CONCLUSIONES FINALES

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

- La elaboración del marco teórico de la investigación mediante el estudio y el análisis de los principales referentes teóricos en los que se sustenta la investigación, facilitó la adquisición de los conocimientos necesarios para la solución.
- La especificación de los requisitos, así como el análisis y diseño del sistema para la gestión de diagnósticos, permitió definir los elementos necesarios de la implementación del mismo.
- El empleo de las herramientas y tecnologías seleccionadas para la implementación de la solución, propició la correspondencia entre los resultados obtenidos y los esperados, lo cual aseguró el nivel de precisión en el análisis y diseño de la aplicación.
- El empleo de la metodología AUP-UCI en su escenario 4, permitió que se trazara una buena planificación de todo el proyecto, ahorrando tiempo y recursos.
- La implementación del componente para la gestión de diagnósticos favoreció la obtención de un sistema funcional, logrando así contribuir al control y planificación del mantenimiento en los vehículos.
- La realización de pruebas de software permitió comprobar el correcto funcionamiento del sistema. Estas pruebas arrojaron resultados satisfactorios en relación al código y el conjunto de interfaces implementadas.

RECOMENDACIONES

Añadir a la opción buscar de la página la elección de buscar también por marca y modelo.

REFERENCIAS BIBLIOGRÁFICAS

- RAMOS CORIA, Daniel Andrés, 2014. Diseño de un Sistema de Monitoreo OBD-II con comunicación GSM. en línea. 29 abril de 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

- Yunier Rodríguez Cruz; María Pinto MolinaII,2010. Evolución, particularidades y carácter informacional de la toma de decisiones organizacionales. [Accedido 20 noviembre 2022].

- Hugo Roig, 2022. Precios crecientes de combustibles en Sudamérica [Accedido 20 noviembre 2022]. Recuperado a partir de: https://www.mentu.com.py/blog/2452/precios-crecientes-de-combustibles-en-sudamerica

- ESPINOSA, L. A. C. 2015. Pruebas en la Nube. In Proceedings of the Taller Horizonte 2020, La Habana, 7 de julio del 2015 2015, A.S. BATISTA ed. Centro Nacional de Calidad de Software.

- IDAE. (2006). Guía para la gestión de combustible en flotas de transporte por carretera. Madrid: IDAE.
- Ramadán Arcos, 2015. Incrementarán inspecciones técnicas a vehículos en la vía. Recuperado a partir de: https://www.granma.cu/cuba/2015-03-29/incrementaran-inspecciones-tecnicas-a-vehiculos-en-la-via
- Ana Julia Marine, 2020. #Internet en #Cuba planes para 2021. Recuperado a partir de: https://www.mincom.gob.cu/es/noticia/internet-en-cuba-planes-para-2021-y-desaf%C3%ADos-con-la-covid-
- Tomado de Prensa Latina,2020. Internet en Cuba: planes para 2021 y desafíos con la Covid-19. Recuperado a partir de: https://www.mincom.gob.cu/es/noticia/internet-en-cuba-planes-para-2021-y-desaf%C3%ADos-con-la-covid-19?fbclid=IwAR1GQYFjD3Bna8kmPACdpHvEJPDFHW66RDE-LUvWNThKe_imCQYidZhnyts
- TENORIO CÓRDOVA, Beatriz de los Ángeles y CORONEL MAGALLANES, Cristian Eduardo, 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

- PRESSMAN, Roger S, 2010b. Ingenieria de software enfoque practico.7ed.Pressman.PDF. *Ingeniería del software, un enfoque práctico*. en línea. 1 enero 2010. [Accedido 12 julio 2022]. Recuperado a partir de: https://www.academia.edu/44770344/Ingenieria_de_software_enfoque_practico_7ed_Pressman_PDF

- ISRAEL CERVANTES ALONSO y SAÚL OSBORN ESPINOSA SOLÍS, 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

- RODRÍGUEZ SÁNCHEZ, Tamara, 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.Universidad de las Ciencias Informáticas.* La Habana: Universidad de las Ciencias Informáticas.
- SOMMERVILLE, Ian, 2011. *Software engineering*. Ninth Edition. ISBN 978-0-13-703515-1.
- Visual Paradigm, 2022. en línea. [Accedido 7 junio 2022]. Recuperado a partir de: https://www.visual-paradigm.com/
- Visual Studio Code, 2022. en línea. [Accedido 9 junio 2022]. Recuperado a partir de: https://code.visualstudio.com/
- LARMAN, Craig, 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. PRENTINCE HALL, México. ISBN 970-17-0261-1.

- Xetid,2020. XETID, empresa cubana líder en el desarrollo de soluciones tecnológicas para la informatización de la sociedad. Recuperado a partir de: https://www.xetid.cu/es/empresa

- Quiroga, Lourdes Aja. Gestión de información, gestión del conocimiento y gestión de la calidad en las organizaciones. [Accedido 10 junio 2022] 2002. [Citado el: 12 de febfrero de 2015.]

- Andrade de Souza,2021. Diagnóstico - Concepto, características y acepciones. Recuperado a partir de: Recuperado a partir de: (https://concepto.de/diagnostico/#ixzz7ksqPkoXE

- ANASTASIO, Meseguer y ENRIQUE, Javier, 2013. Caracterización de los estilos de conducción mediante smartphones, dispositivos obd-ii y redes neuronales. 12 febrero 2013. [Accedido 10

mayo 2022]. Recuperado a partir de: https://riunet.upv.es/handle/10251/21025Accepted: 2013-02-12T07:41:31Z

- Mendez. 2018. uniciencista.gfrodriguez.online. *uniciencista.gfrodriguez.online.* 10 de febrero de 2018. [Accedido 10 mayo 2022]. Recuperado a partir de: http://uniciencista.gfrodriguez.online/2018/02/herramientas-case-principales-usos.html.
- Gaines, Jeff, Boyd, Geraldine y Copley, Della. 2017. Visual Paradigm Online. *Visual Paradigm Online* 2017. [Accedido 10 mayo 2022]. Recuperado a partir de: https://online.visual-paradigm.com/es/features/.
- Bos, Bert. 2019. w3schools. *W3C* 2019. [Accedido 10 mayo 2022]. Recuperado a partir de: https://www.w3.org/Style/CSS/Overview.en.html
- MDN web docs. 2020. developer.mozilla.org. *developer.mozilla.org* 27 de 4 de 2020. . [Accedido 10 mayo 2022]. Recuperado a partir de: https://developer.mozilla.org/es/docs/Web/JavaScript
- Visual Studio Code: Funcionalidades y extensiones, 2018. El Blog de Aitana – Partner Microsoft y Sage en España. en línea. [Accedido 10 mayo 2022]. Recuperado a partir de: https://blog.aitana.es/2018/10/16/visual-studio-code/
- 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
- 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 septiembre 2022]. Recuperado a partir de: http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S1727897X2020000601154&lng=es&nrm=iso&tlng=es
- Universidad de Alicante. 2020. si.ua.es. si.ua.es. 2020. [Accedido 7 septiembre 2022]. Recuperado a partir de: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.
- Dofactory. 2020. www.dofactory.com. *www.dofactory.com.* 2020. [Accedido 7 septiembre 2022]. Recuperado a partir de: https://www.dofactory.com/net/design-patterns.
- Pressman, Roger S, Ph.D. 2010. *Ingenieria de Software. Un enfoque práctico. Septima Edición.* Mexico, D.F : The Me Graw Hill Companies, 2010.

- PMOinformatica.com. 2020. www.pmoinformatica.com. *www.pmoinformatica.com* 2020. [Accedido 10 septiembre 2022]. Recuperado a partir de: http://www.pmoinformatica.com/p/pruebas-de-software.html.
- Yeeply. 2019. www.yeeply.com. *www.yeeply.com.* 2019. [Accedido 10 septiembre 2022]. Recuperado a partir de: https://www.yeeply.com/blog/que-son-pruebas-unitarias/#que
- MySQL. 2013. [Accedido 10 septiembre 2022]. Recuperado a partir de: http://www.mysql.com/why-mysql/white-papers/las-10-razones-principalespara-usar-mysql-como-base-de-datos-integrada/.
- Definición ABC. 2013. [Accedido 10 septiembre 2022]. Recuperado a partir de: http://www.definicionabc.com/tecnologia/mysql.php#ixzz2ll9vdiwf
- ISO 25010, sin fecha. en línea. [Accedido 10 mayo 2022 a]. Recuperado a partir de: https://iso25000.com/index.php/normas-iso-25000/iso-25010

ANEXOS

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?

Anexo 2 Historias de usuarios

Tabla 10: HU Gestionar marca de auto (Fuente: Elaboración propia).

Historia de usuario	
Número: 2	**Número del requisito:** Gestionar marca de auto
Programador: Amanda Ojeda Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 1 hora
Descripción: Descripción: Permitirá insertar, modificar, eliminar y listar las flotas de vehículos. Para registrar una marca de auto se muestra un formulario con una serie de datos que el usuario deberá llenar para completar el registro. • Modelo (Obligatorio. Campo selector. Permite la selección de un modelo). • Transmisión (Obligatorio. Campo selector. Permite la selección de una transmisión). • Estado (Obligatorio. Campo selector. Permite la selección de un estado). • Ubicación (Obligatorio. Permite la selección de una ubicación).	
Observaciones: Al introducir correctamente los datos el sistema redirecciona al usuario al listado de marcas de autos.	

Prototipo de interfaz:

Tabla 11: HU Gestionar modelo de auto (Fuente: Elaboración propia).

Historia de usuario	
Número: 3	**Número del requisito:** Gestionar modelo de auto
Programador: Amanda Ojeda Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 1 hora
Descripción: Descripción: Permitirá insertar, modificar, eliminar y listar las flotas de vehículos. Para registrar un modelo de auto se muestra un formulario con una serie de datos que el usuario deberá llenar para completar el registro. • Marca (Obligatorio. Campo selector. Permite la selección de un modelo). • Transmisión (Obligatorio. Campo selector. Permite la selección de una transmisión). • Estado (Obligatorio. Campo selector. Permite la selección de un estado). • Ubicación (Obligatorio. Permite la selección de una ubicación). • Año de fabricación (Obligatorio. Campo de tipo fecha. Permite la selección de una fecha).	
Observaciones: Al introducir correctamente los datos el sistema redirecciona al usuario al listado de modelos de autos.	
Prototipo de interfaz:	

Tabla 12: HU Gestionar combustible (Fuente: Elaboración propia).

Historia de usuario	
Número: 4	**Número del requisito:** Gestionar combustible
Programador: Amanda Ojeda Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 1 hora
Descripción: Descripción: Permitirá insertar, modificar, eliminar y listar las flotas de vehículos. Para registrar el combustible de un auto se muestra un formulario con una serie de datos que el usuario deberá llenar para completar el registro. • Marca (Obligatorio. Campo selector. Permite la selección de una marca). • Modelo (Obligatorio. Campo selector. Permite la selección de un modelo). • Estado (Obligatorio. Campo selector. Permite la selección de un estado). • Ubicación (Obligatorio. Permite la selección de una ubicación). • Kilometraje (Obligatorio. Permite la selección de los kilómetros recorridos). • Fecha (Obligatorio. Campo de tipo fecha. Permite la selección de una fecha).	
Observaciones: Al introducir correctamente los datos el sistema redirecciona al usuario a las gráficas de consumo de combustible.	
Prototipo de interfaz:	

Tabla 13: HU Gestionar estado (Fuente: Elaboración propia).

Historia de usuario	
Número: 5	**Número del requisito:** Gestionar estado

Programador: Amanda Ojeda Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 1 hora

Descripción:

Descripción: Permitirá insertar, modificar, eliminar y listar las flotas de vehículos. Para registrar el estado de un auto se muestra un formulario con una serie de datos que el usuario deberá llenar para completar el registro.

- Matricula (Obligatorio).
- Marca (Obligatorio. Campo selector. Permite la selección de una marca).
- Modelo (Obligatorio. Campo selector. Permite la selección de un modelo).
- Kilometraje (Obligatorio. Permite la selección de los kilómetros recorridos).

Observaciones: Al introducir correctamente los datos el sistema redirecciona al usuario al listado de vehiculos.

Prototipo de interfaz:

Tabla 14: HU Gestionar chapa (Fuente: Elaboración propia).

Historia de usuario	
Número: 6	**Número del requisito:** Gestionar chapa

Programador: Amanda Ojeda Fernández	**Iteración asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 1 hora
Riesgo de desarrollo: falta de experiencia con el uso de la tecnología	**Tiempo real:** 1 hora

Descripción: Descripción: Permitirá insertar, modificar, eliminar y listar las flotas de vehículos. Para registrar la chapa de un auto se muestra un formulario con una serie de datos que el usuario deberá llenar para completar el registro. • Marca (Obligatorio. Campo selector. Permite la selección de una marca). • Modelo (Obligatorio. Campo selector. Permite la selección de un modelo).
Observaciones: Al introducir correctamente los datos el sistema redirecciona al usuario al listado de vehículos.
Prototipo de interfaz: