FACULTAD DE TECNOLOGÍAS INTERACTIVAS

Sistema web para la gestión de combustibles por el Gobierno Provincial del Poder Popular Artemisa

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

Autor: Ivanis Mirabal Rodríguez
Tutor(es): M. Sc. Henry Raúl González Brito
M. Sc. Yadira Ramírez Rodríguez
Ing. Michael Martínez Acosta

La Habana, octubre de 2024

"Año 66 de la Revolución"

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título ***"Sistema web para la gestión de combustibles por el Gobierno Provincial del Poder Popular Artemisa"*** concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firman la presente a los **18** días del mes de **octubre** del año **2024**.

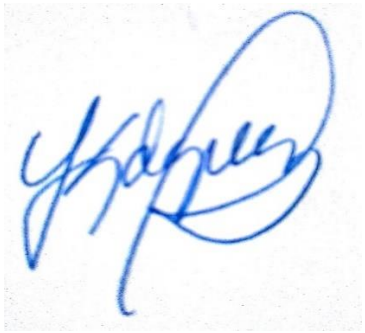
Ivanis Mirabal Rodríguez

_____
Firma del Autor

M. Sc. Henry Raúl González Brito

_____
Firma del Tutor

M. Sc. Yadira Ramírez Rodríguez



_____
Firma del Tutor

Ing. Michael Martínez Acosta

_____
Firma del Tutor

DATOS DE CONTACTO

Ing. Henry Raúl González Brito M. Sc.
Institución: Universidad de las Ciencias Informáticas. Dirección de Seguridad Informática.
Ocupación: Director de Seguridad Informática
Graduado de/Año: Ingeniero Informático/2005.
Correo electrónico: henryraul@uci.cu
Teléfono: 59921416, 53390154 (Whatsapp)

Ingeniero Informático graduado en 2005 en la Universidad de Camagüey y la Universidad Tecnológica de La Habana, Máster en Gestión de Proyectos Informáticos (2012) y diplomados en Inteligencia Tecnológica (2013) y Tecnologías Digitales para la Docencia Universitaria (2023). Actualmente es Profesor Auxiliar y Director de Seguridad Informática en la UCI, además de Coordinador de la Especialidad de Posgrado en Seguridad Informática. Miembro del claustro de varias maestrías. Pertenece al Consejo Científico de la UCI y es miembro de la Junta Directiva Nacional de la Unión de Informáticos de Cuba. Sus investigaciones se centran en seguridad web, ciberseguridad, pruebas de penetración y la superación profesional en este campo. Autor de un libro sobre gestión segura de portales web y reconocido por su labor docente y contribuciones al campo, con premios nacionales e internacionales dentro de las que se destaca la Distinción Especial del MES por su labor de posgrado.

Ing. Yadira Ramírez Rodríguez
Institución: Universidad de las Ciencias Informáticas. Facultad 4.
Graduado de/Año: Ingeniero en Ciencias Informáticas/2007.
Correo electrónico: yramirezr@uci.cu
Teléfono: 58046987

Graduada en el año 2007 de La Universidad de las Ciencias Informáticas (UCI). Posee categoría docente principal de Profesor Auxiliar y es Máster en Calidad de Software. Ha impartido asignaturas de las disciplinas de Práctica Profesional e Ingeniería y Gestión de Software. Actualmente es jefa del Departamento Docente de Informática de la Facultad 4 de

la UCI. Sus líneas de investigación: Ingeniería de Software, Calidad de Software y Gestión de Proyectos.

Ing. Michael Martínez Acosta.
Institución: Gobierno Provincial del Poder Popular Artemisa. Centro de Informática y Comunicaciones.
Ocupación: Jefe de Centro de Informática y Comunicaciones.
Graduado de/Año: Ingeniero Industrial/2017.
Correo electrónico: michael.martinez@gobart.gob.cu
Teléfono: 52793508.
Ingeniero Industrial graduado en 2017 en la Universidad de Artemisa Julio Díaz González y diplomado en Administración Pública (2023). Actualmente es Profesor Instructor. Es Jefe de Centro de Informática y Comunicaciones. Pertenece al Consejo Científico de la Facultad de Ciencias Agropecuarias, Técnicas y Económicas del Centro Universitario Municipal Guanajay perteneciente a la Universidad de Artemisa. Sus investigaciones se centran en el los procesos y servicios, ciberseguridad, la gestión de la calidad, la organización del trabajo y la superación profesional en este campo. Autor de varios artículos en la Revista de Investigación y Ciencia Aplicada a la Ingeniería (INCAING) de México.

AGRADECIMIENTOS

Gracias a todos y en especial a mi hija y mi esposa por darme la fuerza y empuje de llegar hasta donde estoy, obteniendo un título de ingeniera en ciencias informáticas, gracias por ese aliento de confiar en mí cuando todo parecía perdido y no pensaba continuar, tu seguirás, porque sé que si puedes y se los demostraras y a pesar de todos mis tropiezos siempre pude levantarme y seguir adelante porque son ustedes la razón de mi ser, los amos muchísimo.

Al resto de mi familia le agradezco por la preocupación y dedicación que siempre me han dado, siempre reconocer a mis tíos por su apoyo y horas de charla para que continuara y luchara por el sueño que siempre quise, gracias por estar siempre ahí.

Agradecerles a mis tutores, de todo corazón por todas las horas de sueño robadas, consejos, dedicación y empeño para que todo me saliera bien.

Mis compañeros de aula, de verdad que todos me ayudaron mucho, algunos me dieron ánimos cuando no tenía ganas de nada y otros me empujaron para que siguiera.

A todo el que de una forma u otra contribuyó a que esto fuera posible. Gracias

DEDICATORIA

A mis padres, a mi familia y en especial mi hija y mi esposa; motores impulsores de mis logros y los motivos de cada uno de mis sacrificios.

RESUMEN

Actualmente la distribución ineficiente del combustible en la provincia de Artemisa, gestionada mediante métodos manuales y herramientas ofimáticas básicas. La dependencia del uso de Microsoft Excel y la comunicación telefónica generan errores y retrasos en la asignación de combustible a las entidades, afectando la satisfacción de las mismas.

El objetivo de la investigación es desarrollar un sistema de gestión que optimice el control y la distribución del combustible, reduciendo el margen de error humano y mejorando la eficiencia del proceso. Para lograrlo, se aplicaron tecnologías web como Laravel para el marco de trabajo y Bootstrap en el entorno del cliente, integrando bases de datos relacionales para el manejo seguro de la información. Se empleó la metodología de desarrollo ágil XP (Extreme Programming), que permitió un diseño flexible y adaptado a las necesidades del usuario mediante iteraciones cortas y pruebas continuas.

El sistema desarrollado facilitó el registro, seguimiento y distribución de los combustibles, eliminando duplicidades y errores manuales. Como resultado, se cumplió el objetivo propuesto al proporcionar una solución tecnológica que mejora la trazabilidad y eficiencia del proceso de distribución de combustible, reduciendo significativamente el tiempo y esfuerzo dedicado a la tarea.

Palabras clave: combustible, gestión, laravel, sistema de gestión.

ABSTRACT

Currently, the inefficient distribution of fuel in the province of Artemisa is managed through manual methods and basic office tools. The reliance on Microsoft Excel and telephone communication leads to errors and delays in fuel allocation to entities, negatively impacting their satisfaction.

The objective of the research is to develop a management system that optimizes fuel control and distribution, reducing human error and improving process efficiency. To achieve this, web technologies such as Laravel for the framework and Bootstrap for the client-side interface were implemented, integrating relational databases for secure information management. The agile development methodology XP (Extreme Programming) was employed, allowing for a flexible design tailored to user needs through short iterations and continuous testing.

The developed system facilitated the registration, tracking, and distribution of fuel, eliminating duplication and manual errors. As a result, the proposed objective was achieved by providing a technological solution that enhances traceability and efficiency in the fuel distribution process, significantly reducing the time and effort required for the task.

Keywords: fuel, laravel, management, management system.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURA

INTRODUCCIÓN

Las aplicaciones web son esenciales para la transformación digital en la sociedad contemporánea. La interacción entre individuos y organizaciones se realiza predominantemente en el ciberespacio mediante estas aplicaciones. Los usuarios pueden acceder a la web a través de navegadores como Firefox o Chrome, así como mediante aplicaciones móviles nativas que optimizan la presentación de los contenidos para estos dispositivos. Hoy en día, cualquier organización, empresa o entidad que aspire a tener presencia en línea necesita contar con un portal web.

En Cuba, se ha convertido en una herramienta crucial para el desarrollo económico y social del país. En la actualidad esta tarea se lleva a cabo por parte de organismos superiores como el Ministerio de Comunicaciones (MINCOM) entre otros Organismos de la Administración Central del Estado (OACE), con el afán de la creación de servicios en línea para la gestión de trámites, como el pago de impuestos, la solicitud de visas y la consulta de información médica. Además, ha mejorado la comunicación entre empresas y el gobierno, incrementando la eficiencia y la transparencia en la administración de los servicios públicos.

No cabe duda que en los últimos años se han dado cambios sin precedentes en la economía nacional. Cada vez con mayor disposición las administraciones tanto a nivel central del Estado, así como las provinciales y municipales, demandan bienes y calidad en los servicios prestados. Entre las medidas adoptadas por parte del Consejo de Ministros, en el actual modelo económico, aprobado en el VI Congreso del Partido Comunista de Cuba y basado en la Resolución No.1 emitida por el vicepresidente del Consejo de Ministros a los 10 días del mes de diciembre de 2010, se encuentra: “Disolver la estructura del Consejo de la Administración del Poder Popular de La Habana y en consecuencia se aprueba la creación de las Administración Provincial de Artemisa y Mayabeque.” (Jorge, 2010, p. 7).

La presente investigación se realiza en el Centro de Informática y Comunicaciones perteneciente al Gobierno Provincial del Poder Popular Artemisa. Su misión es asegurar la prestación de los servicios de informática a la institución administrativa durante el ejercicio de sus funciones, utilizando la infraestructura disponible y la contratación de servicios recibidos por terceros.

Por consiguiente, para alcanzar sus objetivos en los procesos de trabajo esenciales, el Centro de Informática y Comunicaciones debe: Garantizar la mejor forma de integrar los recursos humanos a los procesos de servicios con la formación del capital humano, lograr un desempeño laboral superior, así como un impacto positivo en la eficiencia, eficacia y productividad de los procesos de implementación de las políticas en la red de gobierno (seguridad informática), la administración y operación en la red de datos, el desarrollo e implementación de aplicaciones informáticas, así como el mantenimiento del equipamiento informático (Cartaya, 2012, p. 185).

Es preciso señalar que Cuba atraviesa un escenario desfavorable en cuanto a la importación de combustibles. En respuesta a esta situación, el miembro del Buró Político y primer ministro, Manuel Marrero Cruz, destacó en una reunión de trabajo efectuada en el Palacio de la Revolución, “la necesidad de fortalecer la integralidad en el enfrentamiento al delito, la corrupción, las ilegalidades y las indisciplinas sociales.” (Díaz & Jesús, 2024).

A lo largo de los años, el gobierno cubano ha promulgado diversas resoluciones para fortalecer y controlar los distintos tipos de combustibles. No obstante, persisten incidentes de personas que vulneran los sistemas de monitoreo y control. Un ejemplo de estos incidentes se puede encontrar en el artículo titulado “Robo de combustible: Fugas y coartadas”, publicado en el sitio web de Cubadebate el 4 de septiembre de 2019 (Figueredo Reinaldo, y otros, 2019).

En el momento actual se precisa como situación problémica que, en la sede del Gobierno Provincial de Artemisa, el Puesto de Dirección tiene la tarea de gestionar la distribución de combustible a diversas entidades y empresas. A pesar de la creciente digitalización, el proceso aún depende en gran medida de herramientas ofimáticas básicas, como Microsoft Excel (versión 2013), lo que ha generado varios problemas operativos y organizativos. El uso de Excel fue una solución rápida para registrar las entradas y salidas de combustible, pero como se realiza manualmente, el proceso es propenso a errores humanos, lo que a menudo resulta en información duplicada, inconsistente o errónea, creando confusiones en las asignaciones de combustible. Además, la necesidad de contabilizar periódicamente las cantidades de combustible distribuidas a cada entidad y llevar un control exhaustivo del saldo disponible en las tarjetas de combustible añade una carga considerable de trabajo a los especialistas. Actualmente, estos deben revisar manualmente documento por documento, lo que prolonga las horas de trabajo y aumenta las posibilidades de cometer errores en los cálculos.

El proceso comienza cuando los servicentros de la provincia comunican sus existencias a través de llamadas telefónicas. Esta información se introduce en Excel, y luego, de forma manual, se rellenan los formularios y modelos correspondientes. A lo largo del día, se reciben múltiples llamadas telefónicas —tanto a líneas fijas como móviles— desde diferentes entidades que buscan ser incluidas en la distribución del combustible. Este procedimiento se realiza dos veces al día (a las 8:00 AM y a las 3:00 PM), y finalmente, la información se envía a los colaboradores de CIMEX mediante fotos enviadas por WhatsApp. Aunque este proceso puede parecer informatizado en cierto grado, es esencial distinguir entre la simple digitalización de la información y una verdadera automatización. Mientras que la digitalización implica el uso de tecnología para registrar datos, la automatización busca optimizar y agilizar los procesos, algo que no se ha logrado en la situación actual.

Este sistema, además de ser altamente complejo, presenta otros problemas críticos. La obtención de información histórica sobre una entidad y las asignaciones previamente realizadas requiere la búsqueda manual de hojas llenadas por diferentes personas, lo que aumenta el riesgo de pérdida o deterioro de documentos, haciendo que la información recopilada no sea completamente precisa ni confiable. Tras recopilar estos datos, los especialistas deben crear nuevos documentos de Excel para realizar las anotaciones y elaborar informes que luego se envían a la gobernación de la provincia. Además, la problemática se agrava por el hecho de que las asignaciones de combustible realizadas por el Gobierno Provincial del Poder Popular de Artemisa a las entidades y empresas resultan insuficientes para cubrir la demanda. Esto genera un clima de insatisfacción tanto en empresas estatales como no estatales, lo que lleva a un incremento en el número de llamadas telefónicas solicitando servicio, aumentando la presión sobre el Puesto de Dirección. La combinación de una insuficiente capacidad de respuesta, el uso excesivo de herramientas inadecuadas para gestionar grandes volúmenes de información y la necesidad de llenar formularios manualmente provoca retrasos, errores en la distribución y una baja satisfacción entre las empresas que dependen del suministro de combustible. Por lo tanto, la problemática radica no solo en la insuficiencia del combustible asignado, sino también en la falta de un sistema adecuado para gestionar de manera eficiente el proceso de distribución, control y seguimiento de las asignaciones de combustible, lo que genera un desgaste innecesario de recursos humanos y

tecnológicos, afectando la capacidad de respuesta y la efectividad del servicio ofrecido por el Gobierno Provincial.

Luego de la situación expresada, se plantea como problema científico:

¿Cómo mejorar la gestión de combustibles por el Gobierno Provincial del Poder Popular de Artemisa a las entidades y empresas en su demarcación?

Lo anterior permite asumir como objeto de la investigación la gestión de combustibles en el país, y como campo de acción la gestión de combustibles en el Gobierno Provincial del Poder Popular de Artemisa.

Se declara como objetivo general de esta investigación:

Desarrollar un sistema informático para mejorar la gestión de combustibles por el Gobierno Provincial del Poder Popular de Artemisa a las entidades y empresas en su demarcación.

Para dar solución al objetivo general, se plantearon las siguientes tareas de investigación:

1. Análisis de los fundamentos teóricos y metodológicos que sustentan la gestión de combustibles por el Gobierno Provincial del Poder Popular de Artemisa a las entidades y empresas en su demarcación, necesarios para la realización de la investigación.
2. Diagnóstico del estado actual de la gestión de combustibles por el Gobierno Provincial del Poder Popular de Artemisa en su demarcación.
3. Implementación de un sistema web para mejorar la gestión de combustibles por el Gobierno Provincial del Poder Popular de Artemisa en su demarcación.
4. Validación del sistema web para resolver el problema de investigación desarrollado.

Para realizar el diagnóstico de la situación actual y la proyección de las soluciones, el autor empleó métodos teóricos y empíricos:

Métodos teóricos:

→ Analítico-Sintético: Se utilizó para la recopilación de información requerida durante la realización del estudio del arte y para el desarrollo del trabajo mediante la revisión de documentos y artículos de donde se extrajeron los elementos más significativos relacionados con los sistemas de asignación de combustibles, además del análisis de las diferentes metodologías, tecnologías y herramientas a utilizar para el desarrollo de la misma.

→ Histórico-lógico: Se empleó en el estudio de cómo ha evolucionado el tema que se está investigando, las diferentes herramientas existentes y sus características, lo que ayuda a la comprensión de la necesidad del sistema.

→ Modelado: Se utilizó para modelar todos los diagramas correspondientes a la etapa de análisis, diseño e implementación del sistema a desarrollar; las relaciones entre objetos y las actividades que intervinieron en el proceso de configuración del entorno colaborativo en la presente investigación.

Métodos empíricos:

→ Entrevista: Se realizaron entrevistas con los trabajadores encargados de llevar el control de esta información, para tener una mejor idea de cómo funcionaba el proceso en ese momento.

→ Observación: Se utilizó para calificar el problema y adquirir lo necesario para desarrollar el sistema de gestión de combustible del Gobierno Provincial del Poder Popular en Artemisa.

La norma bibliográfica que se utilizó fue la APA, siglas de American Psychological Association, por ser un conjunto de reglas y pautas que definen el formato y la estructura de documentos académicos y científicos. Estas normas abarcan desde la organización y presentación del contenido, hasta la citación de fuentes y la creación de una lista de referencias bibliográficas.

La novedad práctica de la investigación radica en la creación de un sistema que mejora las asignaciones de combustible por el Gobierno Provincial del Poder Popular de Artemisa a las entidades y empresas en su demarcación.

La investigación está estructurada en: introducción y tres capítulos, los cuales se relacionan a continuación:

Capítulo 1 - Fundamentos teóricos y metodológicos: En este capítulo, se profundiza en el est ado del arte de la gestión del combustible. Se realizó un estudio bibliográfico exhaustivo sobr e los conceptos clave relacionados con la asignación y control del combustible. Durante este análisis, se seleccionaron las tecnologías de desarrollo más adecuadas para incorporar estas características en el sistema desarrollado. Aquí, se describe el objeto de estudio y el campo de acción de la investigación.

Capítulo 2 - Diseño de la solución propuesta al problema científico: Este capítulo detalla el ca mino seguido durante las etapas de planificación y diseño, siguiendo la metodología XP. Se e

specifican las Historias de Usuarios (HU), el plan de iteraciones y el plan de entregas. Además, se describe la solución propuesta y se mencionan los patrones de diseño utilizados en la implementación de los módulos.

Capítulo 3 - Implementación y prueba de la solución propuesta: En esta etapa, se definen las tareas de ingeniería correspondientes a cada HU. También se aplican pruebas unitarias y de aceptación que permiten verificar el correcto funcionamiento del código y del desarrollo de las funcionalidades implementadas. Se aplicaron pruebas de seguridad para asegurar la robustez del sistema y la protección de la información sensible relacionada con la distribución y consumo de combustibles.

Se incluyen, además, las conclusiones y recomendaciones pertinentes, las referencias bibliográficas y los anexos que sirven de soporte a los temas referidos.

CAPÍTULO I: FUNDAMENTOS TEÓRICOS Y METODOLÓGICOS

En este capítulo se sistematizan los fundamentos teóricos que dan soporte teórico al desarrollo de la investigación relacionados con sistema, sistema web, gestión, framework y hospedaje web. Además, se describen las características que debe tener una plataforma digital aplicada al proceso de gestión de combustibles. Se incluye también un análisis del estado actual de los sistemas informáticos más utilizados para la gestión de combustibles. Conjuntamente, se justifica la elección de la metodología, el lenguaje, las herramientas y las tecnologías utilizadas para el desarrollo de la propuesta de solución.

I.1 Conceptos asociados al dominio de la investigación.

En la actualidad de la era digital, es esencial que cualquier negocio tenga una presencia en línea para alcanzar el éxito. Una de las herramientas más efectivas para lograr esto son los sistemas web. Estos sistemas, que se encuentran en servidores remotos y se acceden mediante navegadores web, proporcionan numerosos beneficios a las empresas, como mejorar la eficiencia operativa y ofrecer una experiencia superior al cliente.

Según Knowdo, los sistemas web o también conocidos como "aplicaciones web" son aquellos que: están creados e instalados no sobre una plataforma o sistemas operativos, sino que se alojan en un servidor en Internet o sobre una intranet (red local). Su aspecto es muy similar a páginas web que pueden ser vistas normalmente, pero en realidad los sistemas web tienen funcionalidades potentes que brindan respuestas a casos particulares, entre los que se encuentras los sistemas de gestión (Knowdo, 2018).

La gestión es la forma en que las empresas organizan y dirigen el flujo de trabajo, las operaciones y los empleados para cumplir con los objetivos de la empresa. El objetivo principal de la gestión es crear un entorno que permita a los empleados trabajar de manera eficiente y productiva. Una estructura organizativa sólida sirve de guía para los trabajadores y establece el tono y el enfoque de su trabajo.

Para Porto la gestión es la “acción y a la consecuencia de administrar o gestionar algo. Gestionar es llevar a cabo diligencias que hacen posible la realización de una actividad. Administrar abarca las ideas de gobernar, disponer, dirigir, ordenar u organizar una determinada cosa o situación.” (Porto, 2021). Recientemente González establece que la gestión es “un proceso que involucra la planificación, organización, dirección y control” (Gonzalez., Concepto de Gestión: Según Autores y Definición, 2024). Afirma Naranjo un sistema de gestión es: Una serie de procesos, acciones y tareas que se llevan a cabo sobre un conjunto de elementos (personas, procedimientos, estrategias, planes, recursos, productos) para lograr el éxito sostenido de una organización, es decir, disponer de los recursos necesarios para satisfacer las necesidades y las expectativas de sus clientes o beneficiarios, trabajadores y de otras partes interesadas a largo plazo y de un modo equilibrado y sostenible (Naranjo, 2015).

Un framework es una estructura o plataforma predefinida que facilita el desarrollo de software. Proporciona un conjunto de herramientas, bibliotecas y buenas prácticas que ayudan a los desarrolladores a crear aplicaciones de manera más eficiente y organizada.

Tiene características principales como: reutilización de código que Permite reutilizar componentes y módulos ya creados, estandarización que ofrece una estructura y convenciones que facilitan el mantenimiento y escalabilidad del proyecto, la productividad acelera el proceso de desarrollo al proporcionar soluciones predefinidas para problemas comunes y su modularidad facilita la integración de nuevas funcionalidades sin afectar el resto del sistema.

Al respecto Gabriela Muente establece que es “una estructura previa que se puede aprovechar para desarrollar un proyecto. El Framework es una especie de plantilla, un esquema conceptual, que simplifica la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo a lo que se quiere realizar.” (Muente, 2020).

Un hospedaje web es como alquilar un espacio en Internet para que tu sitio web pueda estar disponible para que otros lo visiten. Imagina que tienes una tienda física y necesitas un local para que la gente pueda entrar y ver tus productos. El hospedaje web es ese “local” pero en el mundo digital.

Establece Hostinger que un hospedaje web es “un servicio de alojamiento web que te permite publicar un sitio web o aplicación en Internet. Cuando contratas un servicio de hosting, básicamente alquilas un espacio en un servidor físico donde puedes almacenar todos los archivos y datos necesarios para que tu sitio web funcione correctamente.” (Hostinger, 2024).

I.2 Soluciones informáticas para la asignación de combustible

Para el desarrollo de la propuesta de solución se realiza el estudio del arte, este estudio aporta a la investigación la posibilidad de detectar la existencia de soluciones similares para evitar duplicar esfuerzos,

Se definen como principales aspectos a tener en cuenta en la solución propuesta los siguientes:

- **Exportar datos (ED):** formato en que son exportados los datos en caso de ser posible.
- **Almacenamiento de datos (AD):** lugar donde son almacenados los datos del sistema.
- **Gestión de roles y permisos dinámicos (GRP)**: Se ajusta a la necesidad del cliente de poder establecer los permisos necesarios a cada rol ya que es un sistema que como se demostrara es primera vez que va hacer controlado y gestionado por un gobierno.
- **Control de Inventarios (CI)**: Asume la necesidad de registrar las entradas, salidas y existencias de los diferentes tipos de combustibles a gestionar.
- **Existencia en Tarjeta (ET)**: Permite establecer los límites de asignación y solo entregar las cantidades insertadas en las tarjetas.
- **Demanda (D)**: Permite que las entidades o empresas demanden cantidades distintas para cada día.
- **Reportes (R)**: genera reportes con información sobre los inventario o asignaciones.
- **Plataforma (P)**: brindaría una idea general de cómo se desarrollan este tipo de soluciones en el mundo y dentro del país y por qué se implementa en cada caso soluciones de escritorio o de tipo web.
- **Licencia (L)**: tipo de licencia de software.

Sistemas de gestión para la distribución y consumo del combustible a nivel nacional:

1. ServiCupet

Es el nombre de la nueva aplicación que ha creado la UEB de Desarrollo de Software Informático (DSI) de Tecnomática con el objetivo de ayudar a la población a localizar el combustible que necesite en poco tiempo desde cualquier localidad, lo cual le ahorrará combustible extra manejando en busca del mismo, faena que causa frustración a los consumidores porque muchas veces resulta en una gestión fallida.
La apk ya está disponible en APKLIS.cu y descargarla es gratis solo hasta el 31 de diciembre de 2021. Antes de ponerla a disposición de la población, los trabajadores de Tecnomática, CUPET, CUPET-CIMEX, y miembros de la Unión de Informáticos de Cuba (UIC) la usaron y ofrecieron sus opiniones en el grupo público de Telegram TM SERVICUPET APK (t.me/TM_SERVICUPET_APK), al cual pueden acceder todos los ciudadanos que deseen ser testigos de la puesta en marcha de la nueva y útil apk. En cuanto a las especificaciones técnicas, sólo podrán utilizar la aplicación los usuarios de Android desde la versión 4.1 o superior, y no se contempla aún el desarrollo de la misma para los que usan iOS.
La aplicación destaca por su diseño y uso simples y puede trabajar o no con servicios de geolocalización, pero si requiere una conexión a internet, ya sea a través del servicio de red de datos móviles o Wifi para obtener la información actualizada. En ella encuentra la información de contacto de todos los servicentros del país además del tipo de combustible que se vende y que se encuentra en existencia en cada uno de ellos.

2. Apolo

Es un producto nacional perteneciente al Ministerio de las Comunicaciones (MIC) y la Empresa Cubana Nacional de Software (DESOFT), encargada de su desarrollo. Es un sistema cliente-servidor, donde la base de datos está en un servidor central y los usuarios del sistema, independientemente del lugar donde se encuentren, accederán en tiempo real. Es una herramienta enfocada a gestionar todo tipo de flotas de equipos de transporte y ofrecer soluciones de asistencia a la gestión y toma de decisiones, con el objetivo de optimizar la rentabilidad de la inversión mediante la reducción de los costos fijos y variables. Apolo brinda una base informativa, donde la mayoría de la información que se brinda es directamente definible por un usuario de nivel administrativo del sistema (Saavedra, 2013).

Sistemas de gestión para la distribución y consumo del combustible a nivel internacional:

3. CloudFleet

CloudFleet es una aplicación en la nube, diseñada para optimizar tus gastos y aumentar la productividad de tus vehículos. Permite administrar la gran cantidad de información que se genera en este proceso, tales como: tipo de mantenimiento, frecuencia de ejecución de las tareas, distancias recorridas, costos, proveedores, repuestos, almacén, tiempos, mecánicos, entre otros. Además, se integra con CheckList y otros informes de costos e indicadores. Sabemos que el combustible es uno de los rubros más costosos en la gestión de la flota, y por eso hemos desarrollado un módulo que te permitirá analizar detalladamente y desde diferentes puntos de vista su rendimiento y costo. Esta poderosa aplicación te permite realizar listas de chequeo a los vehículos, con el fin de tener el estado real de todas las variables que desee medir y controlar en tu flota. Este módulo te ayuda a administrar de una forma fácil y completa la operación de tus viajes por carretera. También te ayudará a controlar el dinero entregado a los conductores para realizar los viajes. (CloudFleet, 2024)

4. Gestión automatizada de combustible de HID (identiFUEL)

La solución identiFUEL™ consta de un conjunto de componentes fáciles de integrar para sistemas de gestión de combustible (FMS), que permite a los administradores de flotas monitorear y controlar los costos del combustible y la facturación de los vehículos de flotas comerciales. Los FMS se integran sin problemas con bases de datos de clientes que funcionan con software de terceros, reglas y políticas comerciales, interfaces de las bombas de combustible de vehículos y registro y seguimiento de transacciones. Un FMS utiliza los componentes identiFUEL RFID para identificar de forma exclusiva los vehículos autorizados y los conductores, permitiendo así el suministro de combustible de manera segura y controlada.

5. B.SMART

El nuevo sistema de Gestión de Combustible simple e intuitivo creado para PIUSI, consiste en una parte de hardware y una parte de software, incluyendo el APP para los conductores, y un

Software de Nube para los administradores, que gestionarán los surtidores de combustible y las flotas de todos los tamaños. B.SMART se basa en la tecnología Bluetooth para controlar y gestionar fácilmente todas las transacciones desde cualquier dimensión de la flota y desde cualquier lugar. La conexión entre los productos compatibles con PIUSI, las aplicaciones B.SMART y las aplicaciones web es muy simple.

Tabla 1.1: Resultados *del análisis del estudio de homólogos. [Fuente: Elaboración propia]*

Homólogos	**ED**	**AD**	**GRP**	**CI**	**ET**	**D**	**R**	**P**	**L**
ServiCupet	No	Nube	No	No	No	No	No	Móvil	Gratuita / Privativa
Apolo	-	Base de Datos	-	Si	Si	Si	Si	Web	Privativa
cloudFleet	No	Nube	No	Si	No	No	Si	Web	Privativa
identiFUEL	No	Nube	No	Si	No	No	Si	Web / Móvil	Privativa
B.SMART	No	Nube	No	Si	No	No	Si	Web / Móvil	Privativa

A partir del estudio de los sistemas mencionados, se identificaron experiencias positivas que pueden ser implementadas en la investigación. ServiCupet aporta una identidad visual asociada a la Empresa Cubana de Petróleo (CUPET), siendo útil mantener los colores representativos de los distintos tipos de combustibles. En cuanto a B.SMART, se destacó su eficacia en el control de inventarios, mientras que el sistema CloudFleet permitió observar cómo gestionar el control de combustible a través de tarjetas.

Por tanto, los resultados obtenidos a partir de los indicadores definidos fueron:
Existen sistemas que realizan procesos relacionados con el negocio, a pesar de que algunos de estos difieren del proceso que se ejecuta en el gobierno, no se excluyen en su totalidad, ya que presentan varias funcionalidades que pueden ser reutilizadas en la propuesta de solución que se desea. Sin embargo, estos sistemas no incluyen otros elementos significativos del negocio que se desean resolver, como son:
No gestionan los roles y permisos dinámicamente.
No realizan reportes por entidades o empresas en periodos de tiempo.

Luego de realizar un estudio sobre los sistemas de gestión para la distribución y consumo de combustible existentes, se procede a caracterizar la metodología de desarrollo de software seleccionada para la investigación.

I.3 Metodología de desarrollo de software

Algunos autores definen una metodología como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información (Maida & Pacienzia, 2015). Existen dos paradigmas de cuando hablamos de metodologías de software, las tradicionales o robustas y las ágiles. La tabla 1 muestra las principales características de cada una de ellas.

Tabla 1.2. Diferencias entre Metodologías Tradicionales y Agiles. Fuente: Elaboración Propia

Metodologías Tradicionales	**Metodologías Agiles**
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo. Cierta resistencia a los cambios.	Basadas en heurísticas provenientes de prácticas de producción de código. Especialmente preparados para cambios durante el proyecto.
Impuestas externamente. Proceso mucho más controlado, con numerosas políticas/normas.	Impuestas internamente (por el equipo). Proceso menos controlado, con pocos principios.
El cliente interactúa con el equipo de desarrollo mediante reuniones y presenta más artefactos.	El cliente es parte del equipo de desarrollo, presenta pocos artefactos.
Más roles, grupos grandes y posiblemente distribuidos.	Pocos roles, grupos pequeños (menos de 10 integrantes) y trabajando en el mismo sitio.
La arquitectura del software es esencial y se expresa mediante modelos, existe un contrato prefijado	Menos énfasis en la arquitectura del software, no existe contrato tradicional o al menos es bastante flexible.

Según (Yánez Triana, 2022) las metodologías ágiles más usadas hoy en día son Scrum y Extreme Programming (XP). A continuación, se muestra una tabla comparativa desarrollada por el equipo de trabajo que facilita la selección de la metodología ágil que más se ajusta a la investigación según las condiciones actuales de este proceso.

Tabla 1.3. Comparación Scrum, XP y Kanban

Parámetro de calidad	**XP**	**Scrum**	**Kanban**
El cliente forma parte del equipo de desarrollo	Sí	No	No

Aceptar cambios en cualquier momento de alguna iteración	Sí	No	Sí
Programación en pareja	Sí	No	No
Tamaño del proyecto	Bajo-Medio	Medio-Alto	Bajo-Medio
Tamaño del equipo	< 10	<10 y múltiples equipos	<10 y múltiples equipos
Auto organización	No	Sí	Sí

Luego del análisis descrito en la Tabla 1.3, la metodología ágil que más se ajusta a la investigación es XP, pues se adecúa al desarrollo del siguiente trabajo y a los contextos en que se realiza el mismo. El cliente forma parte del equipo de desarrollo del proyecto, logrando una realimentación continua y corrección de errores. A la par que progresa el proyecto, el cliente propone nuevas historias de usuario, modificarlas, o eliminarlas. Permite al equipo la modificación de sus planes en correspondencia con los anteriores. XP al termino de cada iteración exige por la realización de pruebas sobre los requisitos implementados en la iteración planificada para corroborar el cumplimiento de los objetivos propuestos y darles solución a las inconformidades encontradas.

Extreme Programming (XP)

La programación extrema es un método de desarrollo de software dividido en sprints de trabajo. Los marcos ágiles siguen un proceso iterativo, en el que se completa y revisa el marco al final de cada sprint, refinándolo para adaptarlo a los requisitos cambiantes y alcanzar la eficiencia máxima. Al igual que otros métodos ágiles, el diseño de la programación extrema permite a los desarrolladores responder a las solicitudes de los clientes, adaptarse y realizar cambios en tiempo real. Sin embargo, la programación extrema es mucho más disciplinada; realiza revisiones de código frecuentes y pruebas unitarias para realizar cambios rápidamente (Raeburn, 2024).

I.4 Herramientas, tecnologías y lenguajes.

Para escoger las herramientas, tecnologías y lenguajes se revisaron las prestaciones de servicio de la Empresa de Telecomunicaciones de Cuba SA (ETECSA), el servicio seleccionado fue Hospedaje Web, dentro de los requisitos técnicos a cumplir está, utilizar lenguaje de programación Hypertext Preprocessor (PHP) en la versión 7.3.33 y MariaDB como

Sistema Gestor de Base de datos (SGBD). El servicio cumple con parámetros necesarios para el despliegue del Sistema web como son, respaldo energético las 24 Hrs, copia de seguridad diaria de los ficheros y base de datos. Con fin de seleccionar un framework de desarrollo se realizó un estudio entre Laravel, Symfony y Codeigniter donde mediante Google Trends obtuvimos algunas estadísticas (Trends, 2024).

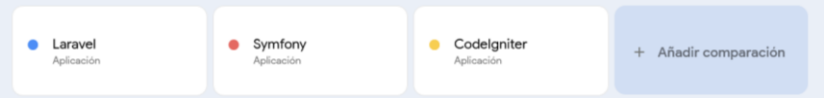


Fig. 1.1: Leyenda de las estadísticas. [Tomado de: (Trends, 2024)]

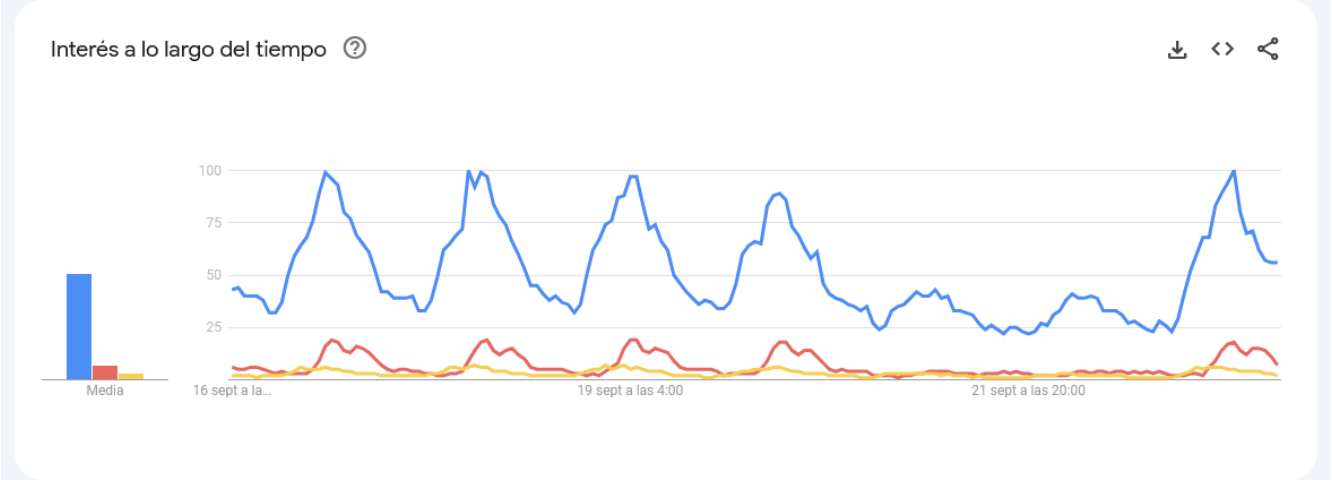


Fig. 1.2: Comparación en el período últimos 7 días, reportado el 23/9/24 [Tomado de: (Trends, 2024)]

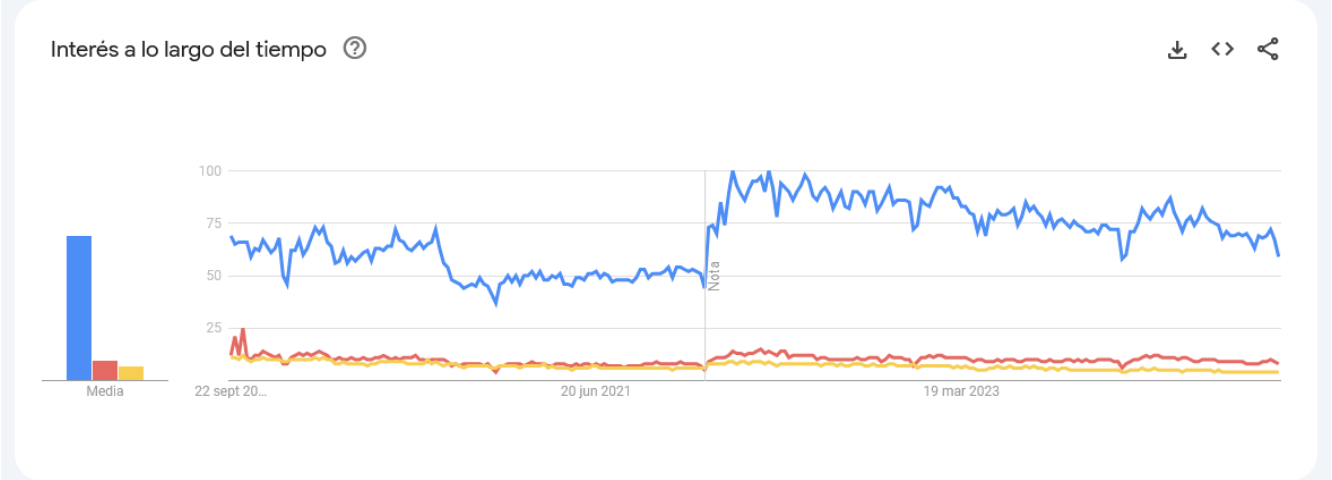


Fig. 1.3: Comparación en el período últimos 5 años, reportado el 23/9/24 [Tomado de: (Trends, 2024)]

Tabla 1.4. Estadística de las FIG 1.2 y 1.3 [Tomado de: (Trends, 2024)]

Framework	Fig. 1	Fig. 2
Laravel	51	69
Symfony	7	10
Codeigniter	3	7

Como se puede apreciar en la Tabla. 1.4, Laravel es un Framework utilizado a nivel mundial y esto permite que la comunidad de desarrollo sea mayor, este aspecto influye sobre todo en equipos pequeños con cortos plazos de desarrollo, por lo que fue la decisión de escoger a Laravel para el desarrollo de la web de asignación de combustible. A continuación, hacemos referencias a las herramientas, tecnologías y lenguajes de programación escogidos y que mejor se relacionan con el framework seleccionado durante la investigación.

I.4.2 Sistema Gestor de Base de datos (SGBD).

Un sistema gestor de bases de datos (SGBD) es un software constituido por una serie de programas dirigidos a crear, gestionar y administrar la información que se encuentra en la base de datos. Su principal objetivo es servir de interfaz entre los usuarios y las aplicaciones para facilitar la organización de los datos, garantizar su accesibilidad, calidad e integridad, brindando a su vez una manera eficaz de administrar esa información. (Universidad Europea, 2021)

MariaDB

MariaDB es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto y gratuito. Es una bifurcación del popular sistema de bases de datos MySQL, y fue creada para ofrecer una alternativa de código abierto a MySQL después de que fue adquirida por Oracle Corporation. MariaDB comparte la mayoría de las características y funcionalidades de MySQL, incluyendo soporte para el lenguaje SQL (*Structured Query Language*), transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), indexación, replicación, escalabilidad y alta disponibilidad. También es compatible con diferentes lenguajes de programación y herramientas de desarrollo. MariaDB tiene algunas características y mejoras únicas, como el soporte para múltiples motores de almacenamiento, incluyendo InnoDB, Aria, MyISAM y más, así como el soporte mejorado para la escalabilidad y la replicación utilizando Galera Cluster.

También es compatible con una amplia variedad de sistemas operativos y plataformas, incluyendo Linux, FreeBSD, Windows y más. (MariaDB Foundation, 2024)

I.4.3 Lenguajes y elementos de programación

Los lenguajes de programación son conjuntos de reglas y sintaxis que permiten a los desarrolladores comunicarse con las computadoras para realizar tareas específicas. Estos lenguajes varían en propósito y características, siendo adecuados para diferentes proyectos. Los elementos de programación, como variables, estructuras de control, funciones y objetos, son componentes esenciales que conforman estos lenguajes y permiten la creación de código eficiente. En este epígrafe, exploraremos los lenguajes más utilizados y sus elementos clave, analizando sus características y cómo influyen en el éxito del desarrollo de software.

Hypertext Preprocessor (PHP)

Es un robusto lenguaje de programación del lado del servidor. Fue diseñado por Rasmus Lerdorf en 1994 como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (*Form Interpreter*) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje (Vaswani, 2010). Dado que es un lenguaje abierto dando la posibilidad de modificar el código fuente y añadir nuevas funcionalidades ha tenido una rápida evolución y desarrollo. Una de sus grandes potencialidades es su soporte para una gran cantidad de bases de datos. De las cuales se pueden mencionar InterBase, MySQL, Oracle y PostgreSQL. PHP también ofrece la integración con varias bibliotecas externas, que dan al desarrollador la posibilidad de realizar cualquier tarea, desde generar documentos en pdf (*Portable Document Format*) hasta analizar código XML.

Estas son algunas de las ventajas que presenta este lenguaje:

- Es un lenguaje multiplataforma.
- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.

- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.

JavaScript

Es un lenguaje de programación interpretado que se utiliza principalmente en el desarrollo de aplicaciones web. Fue creado por Netscape en 1995 y desde entonces ha evolucionado para convertirse en uno de los lenguajes de programación orientado a objeto y basado el prototipos más populares y utilizados del mundo. Se utiliza para crear interactividad en la página web, como la validación de formularios, la creación de animaciones, la manipulación del DOM (*Document Object Model*), la interacción con servidores, entre otros. (Flanagan, 2020) JavaScript es compatible con todos los navegadores web modernos y se ejecuta directamente en el navegador del cliente, por lo que no se necesita ningún software adicional para utilizarlo. Además, se puede utilizar en el servidor a través de plataformas como Node.js.

CCS 3 (Cascading Style Sheets)

Este lenguaje se usa para organizar y controlar la presentación y aspecto de una página Web. Es principalmente utilizado por los diseñadores y programadores de aplicaciones para elegir la multitud de opciones de presentación que este brinda, como son los colores, tipos y tamaños de letra. Se seleccionó la versión 3 de este lenguaje por que trae muchas características nuevas que enriquecen la apariencia de las páginas web, entre los que se encuentran: los bordes redondeados, textos sombreados, sombreados de cajas de texto, múltiples fondos, rotación de textos, animaciones y transiciones.

“Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados “documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes” (Olivares Rodríguez, Aparicio Reytor, & De Armas Rodríguez, 2011).

HTML5 (*Hyper Text Markup Language*)

Lenguaje de Marcas de Hipertexto, es uno de los lenguajes de marcado más utilizado para la construcción de páginas web. Es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la síntesis y coloca instrucciones especiales que no muestra el navegador, aunque si le indica como desplegar el contenido del documento, incluyendo texto imágenes y otros medios soportados. Su versión 5 brinda más facilidades que sus antecesores, lo que permite el desarrollo de la aplicación con una mayor vistosidad, calidad e integración, este es soportado por las versiones actuales de los navegadores Mozilla Firefox, Chrome, Chromiun, Safari e Internet Explorer (Lawson & Sharp, 2011).

SweetAlert2

Hay muchas formas de lanzar notificaciones o alertas en una web. Una de esas formas es usando la función alert de JavaScript, que muestra una caja de alerta con un mensaje que indiquemos. En los tiempos que corren esta es una opción muy poco estética. Con Sweet Alert conseguimos dar a los usuarios notificaciones y alertas de un modo mucho más visual.

Con este plugin jQuery daremos un aspecto a los mensajes que lancemos a los usuarios acorde a las tendencias actuales. Además, tenemos la posibilidad de conFIGr el plugin de muchas formas diferentes (Endeos, 2024).

Datatabale

DataTable es una biblioteca JavaScript de software libre y de código abierto que se utiliza para mostrar datos en tablas interactivas y responsivas en aplicaciones web. Proporciona una solución eficaz y sencilla para trabajar con grandes conjuntos de datos y permitir que los usuarios filtren, clasifiquen, paginen y busquen los datos de manera fácil y rápida. DataTable también ofrece una variedad de opciones de personalización y configuración, lo que te permite ajustar la apariencia y funcionalidad de la tabla según tus necesidades específicas (POWER, 2023).

AdminLTE

AdminLTE es una plantilla de diseño de interfaz de usuario (UI, por sus siglas en inglés) basada en Bootstrap para aplicaciones web de administración. Está diseñada para ser fácil de usar y personalizar, y se utiliza a menudo en proyectos de desarrollo de software para proporcionar una apariencia atractiva y consistente para las interfaces de administración. Incluye una gran cantidad de elementos de diseño, como menús de navegación, barras laterales, formularios, tablas, botones, iconos y mucho más. También incluye un conjunto de widgets y páginas predefinidos, como páginas de inicio, páginas de inicio de sesión y páginas de error, que se pueden personalizar fácilmente para adaptarse a las necesidades del proyecto. Es una plantilla de código abierto y está disponible de forma gratuita en su sitio web oficial. Se puede utilizar en proyectos personales o comerciales sin ningún costo, siempre y cuando se cumplan los términos de la licencia. (Cursos de programación Online, 2022).

Jetstream

Es un scaffolding diseñado especialmente para el framework de PHP. Jetstream ofrece una mesa de trabajo prediseñada para comenzar a desarrollar aplicaciones con Laravel.
Alguna de las funcionalidades que trae consigo Jetstream son las siguientes:

- Verificación por correo electrónico
- Autenticación de dos factores
- Administrador de sesiones
- Soporte de API
- Gestión de equipos
- Entre Otros

Jetstream nos ayuda a tener una base sólida para iniciar nuestros proyectos, acortando así los tiempos de desarrollo. Suponiendo que montar un sistema de usuarios en Laravel con gestor de sesiones y doble factor de inicio de sesión nos tome alrededor de 2-3 días, con Jetstream lo tenemos en menos de 5 minutos, esto es una ventaja muy importante si tenemos múltiples desarrollos (Chávez, 2024).

I.4.4 Marcos de trabajo

Un framework es un esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, una especie de plantilla que sirve como punto de partida

para la organización y desarrollo de software. Generalmente, son usados porque permiten acelerar el trabajo y favorecer que este sea colaborativo, reducir errores y obtener un resultado de calidad. Sirve para acometer un proyecto en menos tiempo, y en el sector de la programación, con un código más limpio y consistente, de manera rápida y eficaz. Ofrece una estructura base que los programadores pueden complementar o modificar según sus objetivos (Unir, 2022).

Laravel 8

Laravel es un marco de trabajo de aplicaciones web PHP de código abierto creado por Taylor Otwell. Utiliza arquitectura Modelo Vista Controlador (MVC) para construir aplicaciones web simples y complejas utilizando el lenguaje de programación PHP. Ofrece un entorno de desarrollo altamente funcional, así como interfaces de línea de comandos intuitivas y expresivas. Posee una nueva interfaz del generador de consultas y establece que la migración anónima es el comportamiento por defecto cuando se ejecuta el comando de migración (Solomon, 2021)

Bootstrap 5.1.3

Bootstrap es un marco de trabajo CSS desarrollado por Twitter en 2010, para estandarizar las herramientas de la compañía. Inicialmente, se llamó Twitter Blueprint y, un poco más tarde, en 2011, se transformó en código abierto y su nombre cambió para Bootstrap, desde entonces fue actualizado varias veces. Una de las características más destacadas es la existencia de muchísima documentación en la red sobre su manejo y muchos blogs especializados en ello. Posee compatibilidad con los principales navegadores web, se integra con las librerías de JavaScript y posee un sistema de cuadrículas que te permite crear el diseño de una web insertando el contenido en bloques o columnas (CASAS, 2019).

Apache ECharts

ECharts, un framework de código abierto, basado en la web y multiplataforma que admite la construcción rápida de la visualización interactiva. La motivación está impulsada por tres objetivos: fácil de usar, ricas interacciones integradas y alto rendimiento. El kernel de ECharts es un conjunto de lenguaje de diseño visual declarativo que personaliza los tipos de gráficos integrados. La arquitectura de transmisión subyacente, junto con un renderizador de gráficos

de alto rendimiento basado en HTML5 Canvas, permite la alta capacidad de expansión y el rendimiento de ECharts (Deqing Li, 2018).

Apache EChartsTM es una herramienta de visualización de JavaScript de código abierto, que puede ejecutarse con fluidez en PC y dispositivos móviles. Es compatible con la mayoría de los navegadores web modernos, por ejemplo, IE9/10/11, Chrome, Firefox, Safari, etc. ECharts depende de ZRender, un motor de renderizado gráfico, para crear gráficos intuitivos, interactivos y altamente personalizables. (EChartsTM)

I.4.5 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado (Integrated Development Environment, IDE) es una herramienta digital utilizada para desarrollar software. Ofrece integración desde los pasos más básicos del desarrollo de software, como escribir su código, depurar o incluso compilar sus aplicaciones en un lenguaje que las computadoras puedan entender (Delgado, 2021). Proporciona acceso a bibliotecas y plantillas preconstruidas para tareas de programación estándar que acelera nuestro flujo de trabajo de desarrollo y aplica las mejores prácticas de una manera fluida y fácil (Geek, 2020).

Visual Studio Code

Visual Studio Code es un editor de texto plano desarrollado por Microsoft totalmente gratuito y de código abierto para ofrecer a los usuarios una herramienta de programación avanzada como alternativa al Bloc de Notas. Está escrito totalmente en Electron, un framework utilizado para unir Chromium y Node.js en forma de aplicación de escritorio. No se caracteriza precisamente por un bajo consumo de memoria, pero es muy sencillo de programar, potente y flexible. Una de las mejores características es IntelliSense, esta función permite resaltar la sintaxis de todo el código fuente y, además, nos permite usar funciones como la de autocompletar, basándose en variables, definiciones y módulos (Velasco, 2021)

I.5 Conclusiones del capítulo

Abordado este capítulo se obtenemos las siguientes conclusiones:

- El análisis del marco teórico sobre el proceso de gestión de combustibles y el estudio de los principales conceptos asociados al problema planteado, permitieron establecer

las bases para el desarrollo de la investigación y conocer las características del objeto de estudio.

- A partir de la caracterización y comparación de varias aplicaciones para la gestión de combustibles, se demuestra que se debe desarrollar una aplicación informática que responda a las necesidades planteadas al inicio de la investigación.

CAPÍTULO II: DISEÑO DE LA SOLUCIÓN PROPUESTA AL PROBLEMA CIENTÍFICO

II.1. Introducción del Capítulo

Este capítulo describe el diseño de la solución propuesta para abordar el problema científico, centrándose en las características clave del dominio de la aplicación y la arquitectura técnica que soportará su desarrollo. Se detalla la propuesta de solución, incluyendo la definición de roles responsables de la administración y seguridad de la web, así como los mecanismos que se utilizarán para asegurar la escalabilidad y mantenibilidad de la aplicación.

Las funcionalidades del sistema se organizan en historias de usuario (HU), las cuales capturan los requerimientos de negocio y permiten priorizar las tareas en función de su impacto y valor. A partir de estas historias, se define un plan de iteraciones, el cual detalla el desarrollo incremental del sistema, permitiendo entregar valor de manera continua. Este proceso de desarrollo sigue una metodología ágil, promoviendo la adaptación a cambios y la entrega oportuna de funcionalidades.

II.2. Propuesta de Solución

Después de evaluar las necesidades de la web y ser elegidas las herramientas apropiadas para su implementación, se procedió a definir los módulos que se deben desarrollar para abordar la problemática planteada. Para una mejor representación se propone la siguiente FIG 2.1.

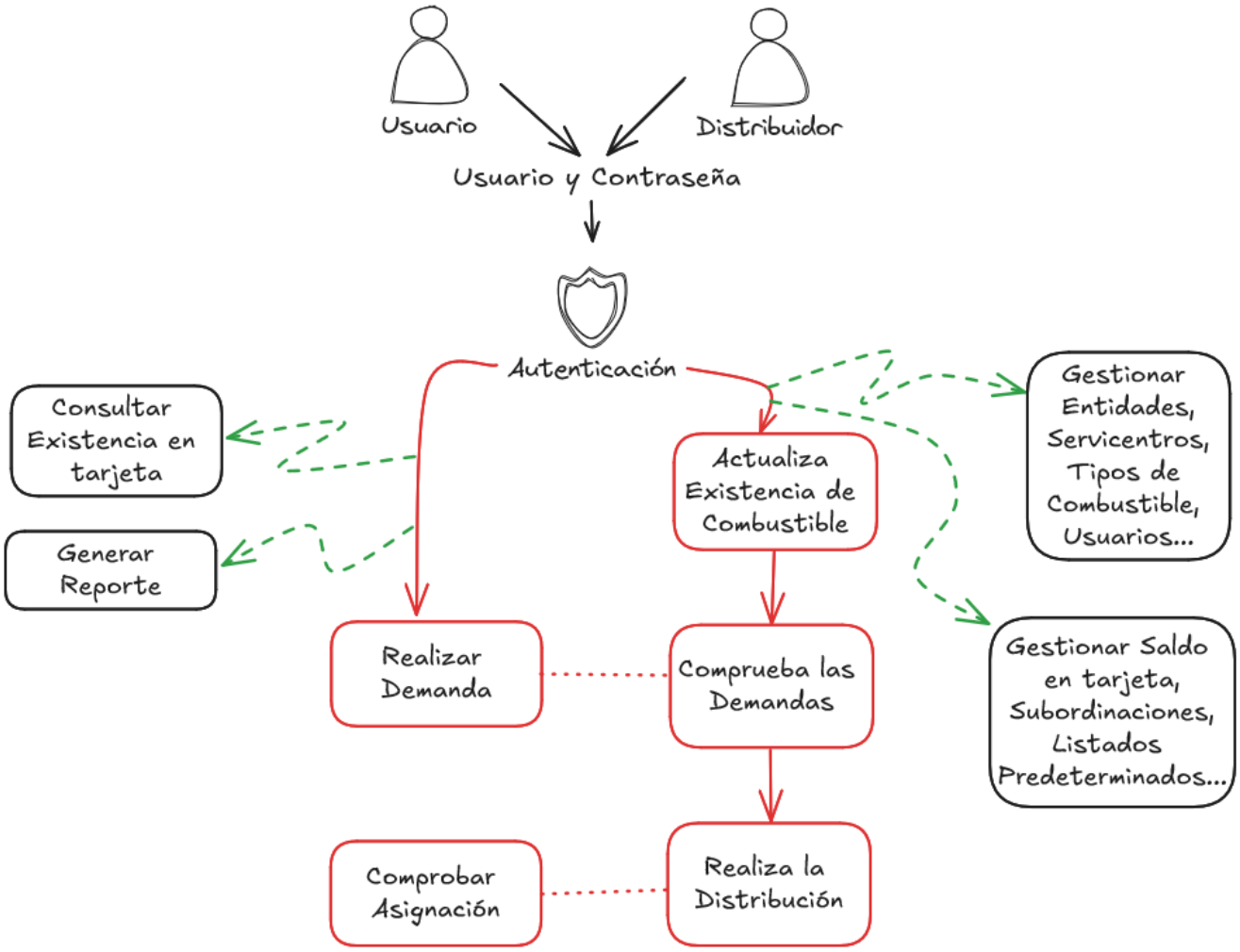


FIG 2.1: Representación gráfica de la web, principales módulos [Toma de: Elaboración Propia]

Módulos que se deben desarrollar para abordar la problemática planteada:

- **Módulo Panel Informativo:** Contiene diferentes reportes sobre las disponibilidades de combustible, entradas en la última semana y un mapa provincial con la densidad por litros de combustibles disponible.
- **Módulo Distribuciones:** Es donde se realiza la entrega de combustible a las entidades o empresas.
- **Módulo Demandas:** En este módulo la entidad o empresa realiza la demanda de la necesidad de combustible para el día.
- **Módulo Existencias:** Aquí se mantiene una actualización en tiempo real sobre lo que va restando por cada tipo de combustible y su CUPET.

- **Módulo Entradas:** Tiene como objetivo guardar cada entrada de combustible al sistema.
- **Módulo Saldo en Tarjeta:** En este módulo se mantiene actualizado el saldo con el que cuentan las entidades o empresas.
- **Módulo Subordinaciones:** Este módulo permite crear una agrupación de entidades o empresas de acorde a la necesidad del cliente.
- **Módulo Listado Predeterminado:** Tiene como propósito crear una lista de distribución por servicentros y tipos de combustibles con las entidades o empresas que le permitirá realizar una distribución en masa.
- **Módulo Nomencladores:** Cuenta con varios apartados, Servicentros, Tipos de Combustible, Entidades, Municipios, Categorías, en todos está permitida la adición, modificación y eliminación de la información.
- **Módulo Usuario:** Tiene como propósito gestionar los usuarios que interactuarán con el sistema.
- **Módulo Roles:** Tiene como propósito gestionar los roles del sistema además de permitir la adición o eliminación de usuarios a ese rol.
- **Módulo Permisos:** Tiene como propósito gestionar los permisos que tendrá un rol en específico.
- **Módulo Trazas de Cambios:** Solo permite visualizar las operaciones realizadas en la aplicación.
- **Módulo Configuración:** Presenta la información de contacto a mostrar a los clientes y define el logo de la aplicación.

II.3. Fase I. Planificación

La etapa inicial del proyecto implica definir el alcance general. Durante esta fase, el cliente explica sus necesidades mediante la creación de HU y les asigna su nivel de prioridad. Después, los programadores calculan el tiempo de desarrollo basándose en esta información. Aunque estas estimaciones son preliminares, ya que se basan en datos generales y podrían variar en cada iteración. También se llega a acuerdos sobre el contenido de la primera entrega y se

establece un cronograma en colaboración con el cliente. Es importante que la primera entrega se consiga en un plazo máximo de tres meses (ESCRIBANO, 2002)

II.3.1. Captura de requisitos.

La captura de los requisitos se refiere a de dónde vienen los requisitos del software y cómo el ingeniero de software puede recogerlos. Es la primera etapa en la construcción de una comprensión del problema que el software quiere solucionar (Bourque, 2004).

II.3.1.1. Listado de requisitos funcionales.

Los requisitos funcionales son especificaciones detalladas de las funcionalidades que un sistema, software o aplicación debe realizar para cumplir con las necesidades de los usuarios o del negocio.

Tabla 2.1 Listado de requisitos Funcionales.

NO	Nombre del requisito	Descripción del requisito	Prioridad
RF1	Crear usuario	El usuario al acceder a la ruta de registro deberá completar los campos nombre, correo, teléfono, cargo, entidad, contraseña y confirmar contraseña.	Alta
RF2	Buscar usuario	El sistema permite la búsqueda de usuario por nombre y entidad.	Media
RF3	Modificar usuario	El sistema permite la modificación de los campos nombre, correo, teléfono, cargo, entidad y el estado del usuario.	Media
RF4	Eliminar usuario	El sistema permite eliminar el usuario.	Alta
RF5	Activar usuario	El sistema permite activar el usuario.	Baja
RF6	Desactivar usuario	El sistema permite desactivar el usuario.	Baja
RF7	Crear rol	Se adiciona un rol solicitando el nombre del rol.	Alta
RF8	Buscar rol	El sistema permite la búsqueda de roles por nombre.	Baja
RF9	Modificar rol	El sistema permite la modificación del campo nombre.	Baja
RF10	Eliminar rol	El sistema permite eliminar el rol.	Baja
RF11	Insertar usuario al rol	Se establece el usuario al rol definido.	Alta
RF12	Eliminar usuario del rol	Se elimina el usuario del rol definido.	Alta
RF13	Crear permiso	El sistema aplica un algoritmo y los permisos son credos según rutas de acceso sean declarados en el sistema.	Alta
RF14	Modificar permiso	El sistema detecta si existe modificación y la refleja en la vista especifica.	Baja

RF15	Eliminar permiso	El sistema detecta si existe eliminación de un permiso.	Baja
RF16	Insertar permiso al rol	Se asigna el permiso al rol.	Alta
RF17	Eliminar permiso del rol	Se elimina el permiso al rol.	Alta
RF18	Crear categoría	El sistema permite la creación de categoría completando el campo nombre.	Baja
RF19	Buscar categoría	El sistema permite buscar categoría por el nombre.	Baja
RF20	Modificar categoría	El sistema permite modificar el campo nombre.	Baja
RF21	Eliminar categoría	El sistema permite eliminar la categoría.	Baja
RF22	Crear municipio	El sistema permite la creación de municipio completando el campo nombre.	Baja
RF23	Buscar municipio	El sistema permite buscar municipio por el nombre.	Baja
RF24	Modificar municipio	El sistema permite modificar el campo nombre.	Baja
RF25	Eliminar municipio	El sistema permite eliminar el municipio.	Baja
RF26	Crear entidad	El sistema permite la creación de entidad completando los campos nombre, municipio y categoría.	Baja
RF27	Buscar entidad	El sistema permite buscar entidad por el nombre, municipio y categoría.	Baja
RF28	Modificar entidad	El sistema permite modificar los campos nombre, municipio y categoría.	Baja
RF29	Eliminar entidad	El sistema permite eliminar el municipio.	Baja
RF30	Crear tipo de combustible	El sistema permite la creación de tipo de combustible completando el campo nombre.	Baja
RF31	Buscar tipo de combustible	El sistema permite buscar tipo de combustible por el nombre.	Baja
RF32	Modificar tipo de combustible	El sistema permite modificar el campo nombre.	Baja
RF33	Eliminar tipo de combustible	El sistema permite eliminar el tipo de combustible.	Baja
RF34	Crear servicentro	El sistema permite la creación de servicentro completando los campos nombre, municipio y tipo de combustible.	Baja
RF35	Buscar servicentro	El sistema permite buscar entidad por el nombre, municipio y municipio y tipo de combustible.	Baja
RF36	Modificar servicentro	El sistema permite modificar los campos nombre, municipio y municipio y tipo de combustible.	Baja

RF37	Eliminar servicentro	El sistema permite eliminar el servicentro.	Baja
RF38	Modificar existencia	El sistema actualiza la existencia del servicentro y el tipo de combustible adicionando la cantidad reflejada en el campo agregar	Alta
RF39	Modificar reserva	El sistema actualiza la existencia del servicentro y el tipo de combustible modificando la cantidad reflejada en el campo agregar	Baja
RF40	Crear entrada	El sistema de forma automática al registrar un incremento en la existencia genera la entrada con los datos fecha, servicentro, tipo de combustible y cantidad.	Media
RF41	Buscar entrada	El sistema permite buscar entrada con los datos fecha, servicentro, municipio, tipo de combustible y cantidad.	Media
RF42	Eliminar entrada	El sistema permite eliminar la entrada.	Baja
RF43	Crear distribución	El sistema para agregar una distribución debe permitir seleccionar una demanda o un servicentro donde va a ser realizada la asignación, se tienen q completar los campos del servicentro, entidad y cantidad.	Alta
RF44	Buscar distribución	El sistema permite buscar una distribución por entidad, municipio, servicentro, tipo de combustible y cantidad.	Alta
RF45	Eliminar distribución	El sistema permite eliminar la distribución.	Alta
RF46	Crear demanda	El sistema permite que el usuario cree la demanda completando los campos nombre de la entidad, las cantidades por tipos de combustibles y la sugerencia de igual forma.	Alta
RF47	Buscar demanda	El sistema permite buscar demanda por entidad, municipio, tipo de combustible y cantidad	Baja
RF48	Eliminar demanda	El sistema permite eliminar la demanda.	Alta
RF50	Crear saldo en tarjeta	El sistema permite seleccionar la entidad que se desea agregar saldo en tarjeta y se completan las cantidades por tipos de combustibles.	Baja
RF51	Buscar saldo en tarjeta	El sistema permite seleccionar la entidad y muestra en una tabla los tipos de combustibles y sus cantidades.	Baja

RF52	Modificar saldo en tarjeta	El sistema permite seleccionar la entidad y modificas las cantidades por tipos de combustibles.	Baja
RF53	Eliminar saldo en tarjeta	El sistema permite eliminar el saldo en tarjeta se modifican los valores a 0 en todos los tipos de combustibles.	Baja
RF54	Crear subordinación	El sistema permite seleccionar la entidad principal que tendrá bajo su subordinación otras entidades y en el campo entidades subordinadas se seleccionan tantas entidades como requiera.	Baja
RF55	Buscar subordinación	El sistema permite seleccionar la entidad deseada y el sistema carga automáticamente las entidades asociadas a esa entidad principal.	Baja
RF56	Modificar subordinación	El sistema permite seleccionar la entidad principal y se eliminan o agregar entidades nuevas a esa subordinación.	Baja
RF57	Eliminar subordinación	El sistema permite escoger la entidad principal y se eliminan todas las relacionadas y queda sin subordinación en el sistema.	Baja
RF58	Reporte de distribución	El sistema permite que al presionar el botón rojo con icono de PDF en la vista distribución y el sistema obtiene la fecha y el horario en el que se esta trabajando y exporta en formato PDF la distribución.	Alta
RF59	Reporte de asignación a entidad	El sistema permite seleccionar el botón verde con icono de filtro y se selecciona el rango de fecha y la entidad sobre la que se necesita saber las asignaciones.	Alta
RF60	Reporte de Existencia de combustible	El sistema permite que en la vista existencia el botón rojo con icono de PDF al presionarlo muestra los servicentros y sus existencias.	Alta
RF61	Reporte de los nomencladores	El sistema permite exportar cada tabla de los nomencladores utilizando una barra en la parte superior de las tablas agregando los distintos tipos de formatos a ser exportados.	Baja
RF62	Crear listado predeterminado	El sistema permite seleccionar el servicentro a los cuales con frecuencia asisten las entidades y en el campo entidades se seleccionan tantas entidades como requiera.	Media

RF63	Buscar listado predeterminado	El sistema permite seleccionar el servicentro deseado y el sistema carga automáticamente las entidades asociadas.	Baja
RF64	Modificar listado predeterminado	El sistema permite seleccionar el servicentro y se eliminan o agregar entidades nuevas.	Baja
RF65	Eliminar listado predeterminado	El sistema permite escoger el servicentro y se eliminan todas las relacionadas y queda sin entidades en el sistema.	Alta
RF66	Agregar distribución por el listado	Al acceder a la vista de distribución por listados se escoge el servicentro, la fecha y el horario de distribución y se les asigna a las entidades antes relacionas en el listado y se agregan las cantidades de acuerdo al tipo de combustible del servicentro.	Media
RF67	Buscar trazas	El sistema permite buscar la traza por IP, usuario, modelo, acción, datos cambiados y nuevos datos.	Alta
RF68	Modificar configuración	El sistema permite modificar los datos del gobierno y datos de contacto.	Baja
RF69	Crear respaldo de base de datos	El sistema permite seleccionar las tablas que se requieran para guardar una copia de la misma.	Alta

II.3.1.2. Listado de requisitos no funcionales.

Los requisitos no funcionales son especificaciones que describen cómo debe comportarse un sistema en términos de calidad, rendimiento, seguridad, y otros atributos que no están directamente relacionados con las funciones específicas del sistema, pero que son igualmente importantes para garantizar su éxito.

- **Software**

Servidor

RNF 1: Servidor web apache 2.4

RNF 2: Motor de base de datos MariaDB 10.4

RNF 3: Versión de PHP 7.3.33

RNF 4: Instaladas las extensiones php-cli y habilitado el modo rewrite permitiendo como fichero de seguridad .htaccess

Cliente

RNF 5: Navegador web en cualquier dispositivo.

- **Hardware**

El sistema debe tener para el servidor:

RNF 6: CPU de 4 núcleos a 2.5 GHz

RNF 7: RAM: 2 GB

RNF 8: HDD: 40 GB

RNF 9: Tarjeta de red: 100 Mb/s

El sistema debe tener para el cliente:

RNF 10: CPU de 2 núcleos a 1 GHz

RNF 11: RAM: 1 GB

RNF 12: HDD: 20 GB

RNF 13: Tarjeta de red: 100 Mb/s

- **Usabilidad**

RNF 14: El sistema web está enfocado a personas con nivel técnico superior, debe permitir realizar las tareas principales (registrarse, iniciar sesión, demandar y consultar asignaciones) en menos de 5 pasos y sin necesidad de consultar ayuda externa.

RNF 15: Se mantienen los colores de la identidad visual de CUPET para identificar los colores de los dispensadores de combustibles como son el color rojo para la gasolina B90, azul la gasolina B94, gasolina B83 de color negro y el color amarillo el Diesel.

RNF 16: Se utilizará el color rojo y negro en mayor medida para representar la provincia de Artemisa donde se realiza la actual investigación.

- **Seguridad**

RNF 17: Los roles de cada usuario serán definidos por el administrador, serán establecidos según las funcionalidades que podrá realizar cada uno, garantizando que la información no sea expuesta a personal indebido.

RNF 18: El sistema deberá mostrar mensajes de error genérico, no especificándole al usuario donde está el problema, evitando exponer puntos críticos del sistema.

RNF 19: Los usuarios deberán después realizar el registro esperar por la aprobación de los usuarios autorizados a permitirles el acceso a otros usuarios de nueva creación.

RNF 20: El servidor contará con un certificado de seguridad generado con Let's Encrypt permitiendo que el navegador no muestre excepciones de que el certificado no es válido.

RNF 21: El fichero .htaccess deberá contener las siguientes instrucciones para minimizar las vulnerabilidades.

```
<FilesMatch "^\.env$">
    Order Allow,Deny
    Deny from all
</FilesMatch>
<Files "laravel.log">
    Order Allow,Deny
    Deny from all
</Files>
<IfModule reqtimeout_module>
         RequestReadTimeout    header=20-40,MinRate=500
body=20,MinRate=500
</IfModule>
<FilesMatch "\.map$">
    Order Allow,Deny
    Deny from all
</FilesMatch>
```

- **Rendimiento**

RNF 22: El sistema debe ser capaz de recibir más de 40 peticiones simultaneas.

RNF 23: El sistema debe responder con una carga de página entre dos segundos y cinco segundos.

RNF 24: Uso de memoria inferior a 500 MB en operaciones normales

II.3.2. Historias de Usuarios.

Las HU son una técnica empleada por la metodología XP para definir los requisitos del software. En el contexto del Proceso Unificado (UP), las HU son equivalentes a los casos de

uso y desempeñan un papel fundamental en la especificación de funcionalidades. Se presentan como tarjetas de papel en las que el cliente describe de manera concisa las características que el sistema debe tener.

En las HU se considera:

III.1. La prioridad en el negocio:

- **Muy Alta:** Esencial para el funcionamiento del sistema según los clientes.
- **Alta:** Necesaria para el sistema y afecta directamente su desarrollo.
- **Media:** Considerada necesaria, pero no crucial para el flujo principal del sistema.
- **Baja:** No afecta significativamente el flujo principal del sistema.

III.2. El riesgo en desarrollo:

- **Alto:** Puede llevar a errores críticos que inutilicen el código.
- **Medio:** Puede retrasar la entrega del sistema.
- **Bajo:** Errores manejables sin perjuicio importante para el sistema.

Teniendo en cuenta el uso del framework de desarrollo Laravel se decide agrupar las funcionalidades y describirlas como gestionar, pues la programación se realiza de manera sencilla y prácticamente con un esfuerzo mínimo teniendo en cuenta las funcionalidades que ofrece dicho framework, además se establece que una semana contará con cinco días laborables, dejando de descanso dos días, sábado y domingo, cada día laborar contará con ocho horas de trabajo.

Tabla 2.2. Historia de Usuario #8. Gestionar Servicentro.

Historias de Usuarios	
Número: HU_8	**Nombre del Requisito**: Gestionar Servicentro
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.4
Prioridad: Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de los servicentros es crucial en el sistema. Este nomenclador alimenta otros módulos y permite modificar y eliminar datos desde la columna de opciones. El botón de "Adicionar" está en la parte superior. • **Mostrar:** Visualiza todos los servicentros con sus columnas de nombre, municipio, combustibles, disponibilidad y opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Adicionar:** Permite ingresar nombre, municipio, tipos de combustibles y disponibilidad. • **Modificar:** Permite cambiar todos los datos previamente ingresados. • **Eliminar:** Permite eliminar servicentros seleccionando nombre, municipio, tipos de combustibles y disponibilidad.	
Prototipo de Interfaz:	

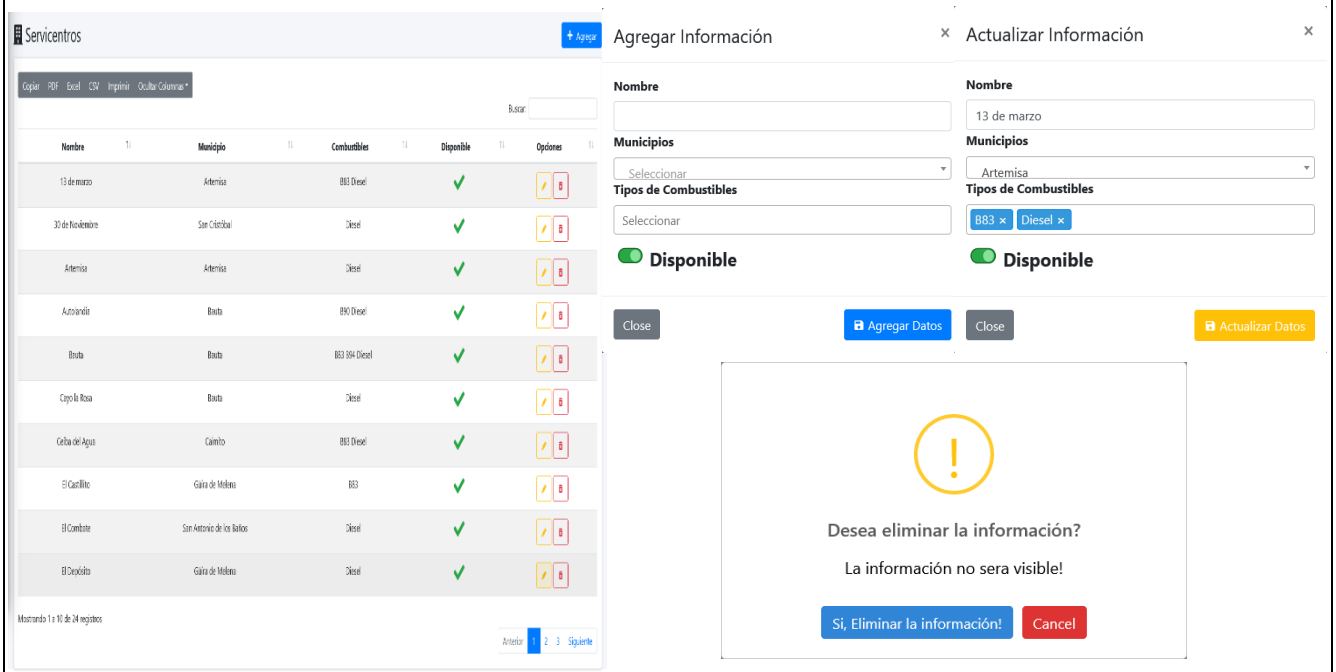


Tabla 2.3. Historia de Usuario #9. Gestionar Existencia.

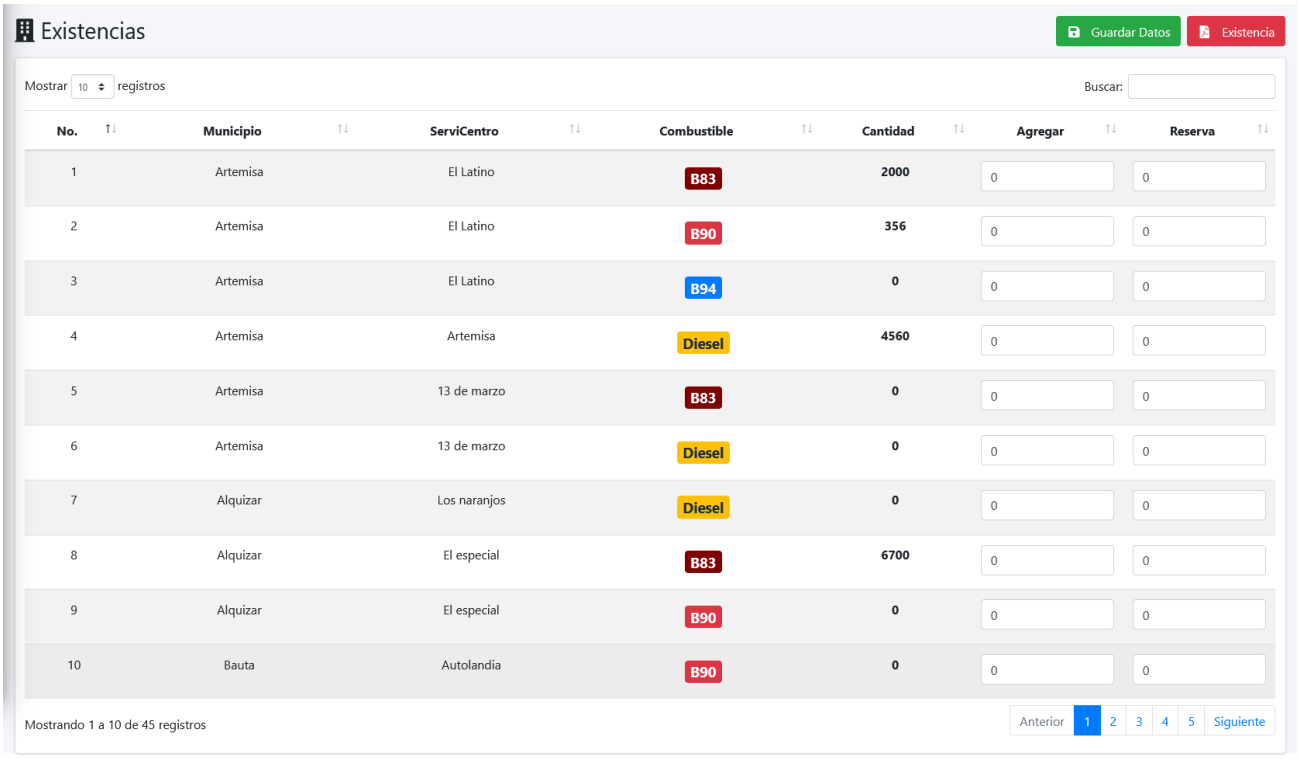
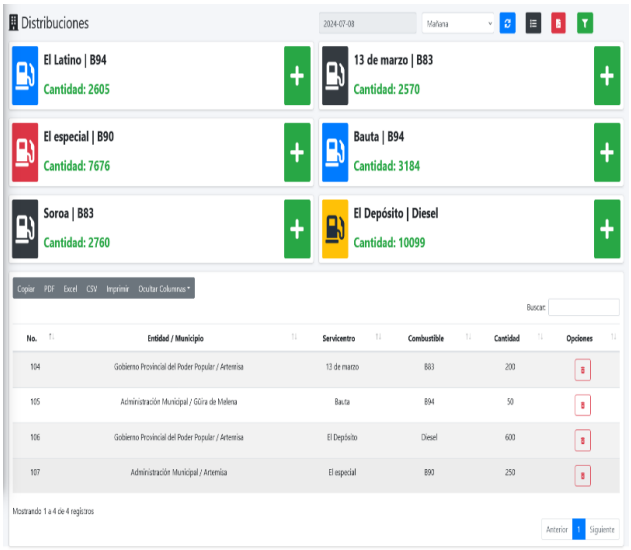
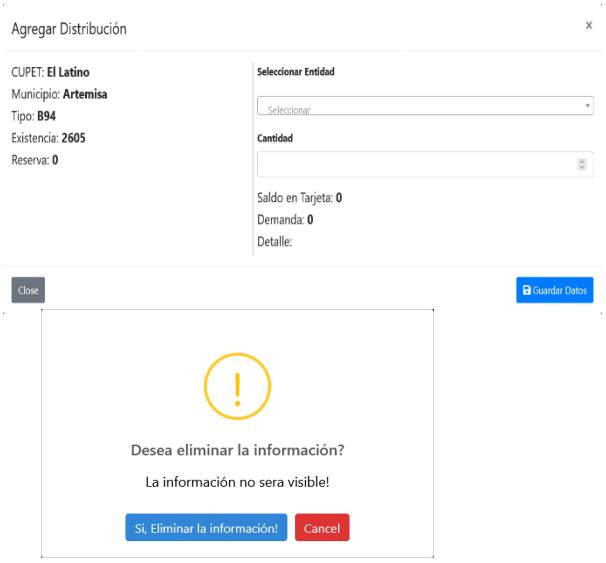
Historias de Usuarios	
Número: HU_9	**Nombre del Requisito**: Gestionar Existencia
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 1.0
Prioridad: Muy Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: Se visualiza un listado de todos los servicentros y sus combustibles con la cantidad correspondiente. Permite adicionar y ajustar existencias. Las entradas se registran, pero los ajustes negativos no se consideran como entradas debido a variaciones técnicas en los niveles de combustible. • **Agregar Entrada:** Escribe el dato en la celda del servicentro. Si es negativo, solo ajusta la existencia; si es positivo, se registra como entrada y se suma a la existencia previa. • **Agregar Reserva:** Escribe el dato en la celda del servicentro. Si es negativo, ajusta la reserva; si es positivo, actualiza la información. Esta actualización no es acumulativa.	
Prototipo de Interfaz: 	

Tabla 2.4. Historia de Usuario #11. Gestionar Distribución.

Historias de Usuarios	
Número: HU_11	**Nombre del Requisito**: Gestionar Distribución
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 2.0
Prioridad: Muy Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de la distribución es esencial para el sistema. Este módulo asigna combustibles a entidades, ajusta la existencia en servicentros y actualiza los datos en las tarjetas de las entidades. Permite seleccionar horario y día de distribución. Para facilitar la asignación en la mañana, se introduce un botón que recoge la información del módulo de Listado Predeterminado. Muestra servicentros con disponibilidad de combustible mayor a 0 y un listado de lo asignado. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Agregar:** Al presionar el botón de asignación del servicentro, se abre un modal con detalles del servicentro y permite seleccionar la entidad y cantidad de combustible a asignar. Muestra la existencia en tarjeta y demanda para la fecha y horario seleccionados. • **Eliminar:** Muestra una alerta de eliminación y, si se confirma, devuelve al servicentro y a la tarjeta la cantidad asignada. • **Agregar por Listado:** Redirige a una nueva página para seleccionar servicentro, fecha y horario. Permite ver la existencia disponible por tipos de combustibles y las entidades seleccionadas para abastecer. Después de insertar los datos, se presiona el botón de guardar.	
Prototipo de Interfaz:   	

II.3.3. Estimación de esfuerzo por Historia de Usuario

Con la finalidad de obtener un correcto desarrollo del sistema se efectúa una estimación del esfuerzo requerido para la implementación de cada historia de usuario.

II.3.4. Desarrollo del plan de iteraciones

Durante la etapa de desarrollo del sistema se crea el plan de iteraciones con el objetivo de especificar la prioridad con que se implementaran las historias de usuario teniendo en cuenta para ello el esfuerzo asociado y las prioridades definidas por el cliente.

Tabla 2.5. Estimación de esfuerzo por Historia de Usuario.

#	Historia de Usuario	Puntos estimados (semanas)
1	Gestionar Usuario.	0.3
2	Gestionar Rol.	1
3	Gestionar Permisos.	1
4	Gestionar Categoría.	0.3
5	Gestionar Municipio.	0.3
6	Gestionar Entidad.	0.3
7	Gestionar Tipo de Combustible.	0.3
8	Gestionar Servicentro.	0.4
9	Gestionar Existencia.	1
10	Gestionar Entrada.	0.3
11	Gestionar Distribución.	2
12	Gestionar Demanda.	1
13	Gestionar Saldo en Tarjeta.	0.3
14	Gestionar Subordinaciones.	0.3
15	Generar Reportes.	1
16	Gestionar Notificaciones.	0.4
17	Gestionar Listado Predeterminado.	0.3
18	Visualizar Trazas.	0.3
19	Gestionar Configuración.	0.4
20	Respaldar Base de Datos.	0.4

II.3.5. Plan de duración de las iteraciones

En el plan de duración de las iteraciones expuesto a continuación se muestra el valor de la duración en semanas de cada iteración y el orden en el que van a ser implementadas las historias de usuario de cada iteración.

Tabla 2.6. Plan de duración de las iteraciones.

Iteración	HU	Historia de Usuario	Puntos estimados (semanas)
1	1	Gestionar Usuario	7.1
	2	Gestionar Rol	
	3	Gestionar Permisos	
	4	Gestionar Categoría	
	5	Gestionar Municipio	
	6	Gestionar Entidad	
	7	Gestionar Tipo de Combustible	
	8	Gestionar Servicentro	
	9	Gestionar Existencia	
	10	Gestionar Entrada	
	11	Gestionar Distribución	
2	12	Gestionar Demanda.	2.3
	13	Gestionar Saldo en Tarjeta.	
	14	Gestionar Subordinaciones.	
	15	Generar Reportes.	
3	16	Gestionar Notificaciones.	1.4
	17	Gestionar Listado Predeterminado.	
	18	Visualizar Trazas.	
	19	Gestionar Configuración.	
	20	Respaldar Base de Datos.	
Total			11.3

II.3.6. Plan de entregas

A continuación, se muestra una propuesta de entrega de versiones del sistema. Cada versión se conformará al finalizar una iteración.

Tabla 2.7. Plan de entregas de las iteraciones.

Entregable	Iteración 1	Iteración 2	Iteración 3
Aplicación	Versión 1.1	Versión 1.2	Versión 1.3
Fecha	30/07/2024	16/08/2024	29/08/2024

II.4. Fase II. Diseño del Sistema

Aquí se establece una arquitectura sólida y adaptable que facilita el desarrollo iterativo y la respuesta a cambios. Esta fase incluye la representación arquitectónica, donde se define la

estructura general del sistema, asegurando que todos los componentes funcionen de manera coherente. Además, se aplican patrones de diseño para resolver problemas comunes de manera eficiente; entre ellos, los patrones GRASP (*General Responsibility Assignment Software Patterns*) que ayudan a asignar responsabilidades de manera efectiva, y los patrones GoF (*Gang of Four*) que proporcionan soluciones probadas para problemas de diseño recurrentes. Finalmente, se desarrolla un modelo de datos que representa la estructura y las relaciones de los datos dentro del sistema, garantizando su integridad y accesibilidad.

II.4.1 Representación arquitectónica

Para desarrollar la propuesta de solución se utilizó el patrón de arquitectura cliente-servidor, que permite procesar la información de un modo distribuido entre los componentes del sistema. Brinda la posibilidad de que cada cliente acceda a los recursos de la información con independencia de su ubicación geográfica, plataforma de software o hardware. (FIG 2.1).

FIG 2.2. Esquema de ejecución de las diferentes partes de una aplicación web. [Tomado de: (Invarato, 2019)]

El Cliente

El cliente inicia la comunicación con el servidor enviando solicitudes y recibiendo una respuesta para cada una. En este sistema, los clientes son el navegador web y la aplicación Android, diseñados para facilitar la interacción con el usuario final. El protocolo de comunicación utili-

zado para el intercambio de información es HTTP. El navegador accede a las interfaces proporcionadas por el servidor a través de una dirección URL. La aplicación Android se comunica con una API REST en el servidor, utilizando la biblioteca OkHttp para enviar y recibir solicitudes HTTP.

El Servidor

El servidor es responsable de gestionar la lógica de negocio del sistema y controlar el acceso a los datos por parte de las aplicaciones cliente. Está compuesto por un servidor de aplicaciones web y un servidor de base de datos. Esta API actúa como una capa de abstracción para facilitar el consumo de los servicios ofrecidos por el servidor.

Para desarrollar la propuesta de solución se utilizó el patrón de arquitectura Modelo-Vista-Controlador (MVC), permite una organización clara y estructurada del código en aplicaciones, especialmente en aplicaciones web. Se muestra a continuación una representación gráfica. (FIG 2.3).

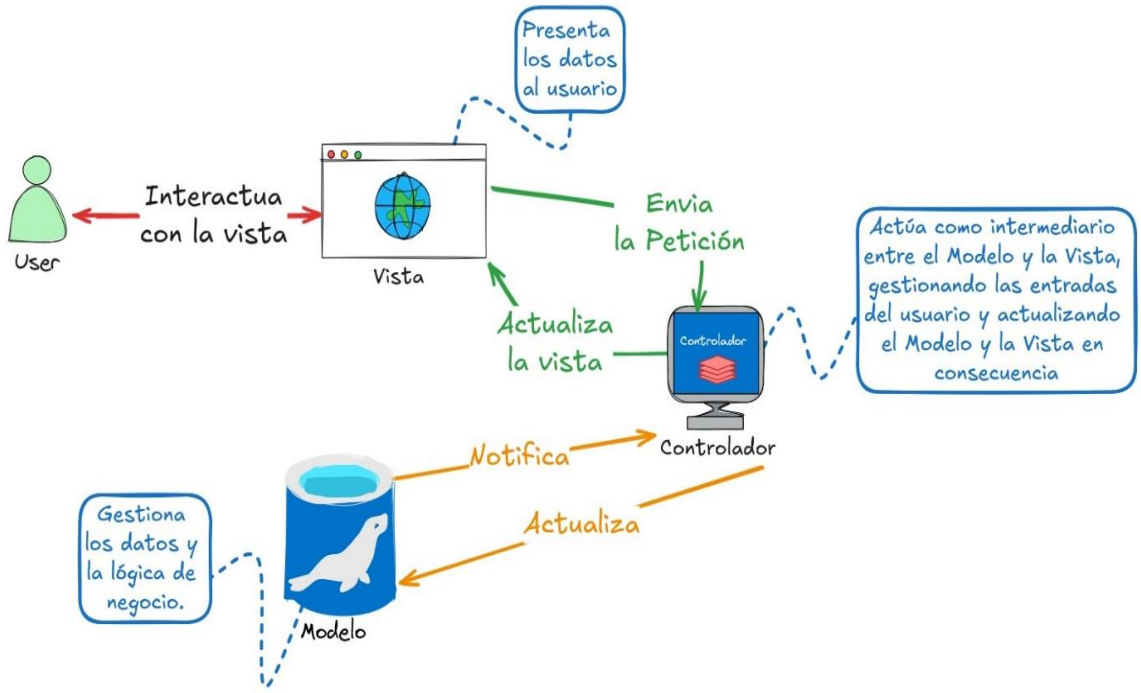


FIG 2.3. Patrón de arquitectura de software Modelo – Vista – Controlador. [Tomado de: Elaboración Propia.]

Componentes del MVC:

1. **Modelo**:
    - **Función**: Gestiona los datos y la lógica de negocio de la aplicación. Es responsable de acceder a la base de datos, realizar cálculos y cualquier otra operación relacionada con los datos.
    - **En Laravel**: Los modelos en Laravel se crean utilizando Eloquent ORM, que permite interactuar con las tablas de la base de datos como si fueran objetos.
    - **Ejemplo**:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use OwenIt\Auditing\Contracts\Auditable;

class NCategoria extends Model implements Auditable
{
    use HasFactory;
    use \OwenIt\Auditing\Auditable;

    protected $fillable = [
        'nombre',
    ];

    protected $hidden = [
        'created_at',
        'updated_at',
    ];
}
```

FIG 2.4. Modelo NCategoria. Fuente: Elaboración propia.

2. **Vista**:
    - **Función**: Presenta la información al usuario. Es responsable de la presentación y la interfaz de usuario.
    - **En Laravel**: Las vistas en Laravel se crean utilizando Blade, un motor de plantillas que permite mezclar PHP con HTML de manera eficiente.

- **Ejemplo**:

```
@extends('adminlte::page')

@section('title', 'Dashboard')

@section('content_header')
<style>···
</style>
  @if (session('permission'))
  <div class="alert alert-danger alert-dismissible">
    <button type="button" class="close" data-dismiss="alert" aria-hidden="true">×</button>
    <h5><i class="icon fas fa-ban"></i> Alerta!</h5>
    {{ session('permission') }}
  </div>
  @endif
  <div class="row">
    @foreach ($combustibles as $comb)
      <div class="col-3">
          <div class="card" id="{{$comb->nombre}}" style="width: 100%; height: 200px;"></div>
      </div>
    @endforeach
    <div class="col-7">
        <div class="card" id="entradas" style="width: 100%; height: 350px;"></div>
    </div>
    <div class="col-5">
      <div class="card" id="mapa" style="width: 100%; height: 300px;"></div>
    </div>
  </div>
  <div id="tooltip" class="custom-tooltip"></div>
@stop
```

FIG 2.5. Vista Panel Informativo. Fuente: Elaboración propia.

3. **Controlador**:
   - **Función**: Gestiona la comunicación entre el modelo y la vista. Recibe las solicitudes del usuario, interactúa con el modelo para obtener los datos necesarios y pasa esos datos a la vista para su presentación.
   - **En Laravel**: Los controladores en Laravel se crean en la carpeta app/Http/Controllers y se utilizan para definir la lógica que maneja las solicitudes HTTP.
   - **Ejemplo**:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\NCategoria;
use Illuminate\Validation\Rule;
use Illuminate\Support\Facades\Validator;

class NCategoriaController extends Controller
{
    public function index()
    {
        return view('nomencladores/categorias');
    }

    public function listar() {
        return NCategoria::All();
    }
```

FIG 2.6. Controlador NCategoriaController. Fuente: Elaboración propia.

II.4.2 Patrones de diseño

Un patrón de diseño se caracteriza como “una regla de tres partes que expresa una relación entre cierto contexto, un problema y una solución” [Ale79]. Para el diseño de software, el contexto permite al lector entender el ambiente en el que reside el problema y qué solución sería apropiada en dicho ambiente. Un conjunto de requerimientos, incluidas limitaciones y restricciones, actúan como sistema de fuerzas que influyen en la manera en la que puede interpretarse el problema en este contexto y en cómo podría aplicarse con eficacia la solución. (Pressman, 2010)

II.4.2.1 Patrones GRASP (*Patrones De Asignación De Responsabilidades*).

Los patrones GRASP nos dan unos principios generales para asignar responsabilidades y se utiliza sobre todo en la realización de diagramas de interacción. Básicamente las responsabilidades de un objeto son conocer y hacer. Dichas responsabilidades implicarán más o menos métodos y clases.

Principales patrones GRASP:

1. **Experto:** Es el que tienen la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada.

```
$existencias = Existencia::with('datos')
                    ->whereHas('datos.servicentro', function ($query) {
                        $query->where('estado', 'on');
                    })
                    ->where('cantidad','!=',0)
                    ->get();
```

FIG 2.7. Ejemplo de Experto. [Tomado de: Elaboración propia]

2. **Creador:** Asigna la responsabilidad a un objeto intermedio para que medie entre otros componentes o servicios, y éstos no terminen directamente acoplados. El intermediario crea una indirección entre el resto de los componentes o servicios. El patrón Creador

guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos.

```
$existencia = Existencia::with(['datos','datos.servicentro','datos.servicentro.municipio','datos.tipo'])
                    ->whereHas('datos', function ($query) use ($request) {
                        $query->where('id', $request->id);
                    })
                    ->first();
```

FIG 2.8. Ejemplo de Creador. [Tomado de: Elaboración propia]

3. **Alta Cohesión:** Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto identificable.

En Laravel este principio se promueve a través del uso de Eloquent ORM donde cada modelo representa una tabla en la base de datos y contiene las operaciones relacionadas con esa tabla. Ejemplo FIG 2.9

4. **Bajo Acoplamiento:** Debe haber pocas dependencias entre las clases.

Este principio se refleja en la forma en que Laravel maneja las dependencias a través de su contenedor de servicios e inyección de dependencias permitiendo que las clases sean más independientes y fáciles de modificar o reemplazar. Ejemplo FIG 2.9

```php
class Distribucion extends Model implements Auditable
{
    use HasFactory;
    use \OwenIt\Auditing\Auditable;

    protected $fillable = [
        'fecha',
        'horario',
        'id_servi_comb',
        'id_entidad',
        'cantidad',
    ];

    protected $hidden = [
        'created_at',
        'updated_at',
    ];

    public function sc(){
        return $this->belongsTo(NrSerComb::class, 'id_servi_comb');
    }

    public function entidad(){
        return $this->belongsTo(NEntidad::class, 'id_entidad');
    }

}
```

FIG 2.9. Modelo Distribución. [Tomado de: Elaboración propia]

II.4.2.2 Patrones GoF.

Los patrones "Gang of Four" (GoF) son un conjunto de patrones de diseño de software que fueron documentados por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides en su libro "Design Patterns: Elements of Reusable Object-Oriented Software". Estos patrones representan soluciones probadas para problemas comunes en el diseño de software orientado a objetos.

Los patrones "Gang of Four" se dividen en tres categorías principales:

1. **Patrones de creación:** Estos patrones se centran en la creación de objetos de manera que se pueda encapsular este proceso, facilitando la flexibilidad en la creación de instancias.

```
class User extends Authenticatable implements Auditable, JWTSubject
{
    use HasFactory;
}
```

FIG 2.10. Ejemplo de creación. [Tomado de: Elaboración propia]

2. **Patrones de estructura:** Estos patrones se ocupan de la composición de clases y objetos para formar estructuras más grandes, proporcionando formas efectivas de manejar las relaciones entre entidades.

```
use HasApiTokens;
use HasFactory;
use HasProfilePhoto;
use Notifiable;
use TwoFactorAuthenticatable;
use HasRoles;
```

FIG 2.11. Ejemplo de estructura. [Tomado de: Elaboración propia]

3. **Patrones de comportamiento:** Estos patrones se centran en la comunicación entre objetos, definiendo cómo los objetos interactúan entre sí y cómo se distribuyen las responsabilidades.

```
use OwenIt\Auditing\Contracts\Auditable;
use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Authenticatable implements Auditable, JWTSubject
{
    use \OwenIt\Auditing\Auditable;
}
```

FIG 2.12. Ejemplo de comportamiento. [Tomado de: Elaboración propia]

Singleton: Solamente permite a una clase u objeto tener una sola instancia y usa una variable global para almacenar esa instancia. Puedes usar carga perezosa para asegurar que hay solamente una instancia de la clase, porque creará la clase solamente cuando lo necesites. (McGregor, 2023)

```php
public function adminlte_image()
{
    return (Auth::user()->profile_photo_path)
        ? asset('public/storage/usuario/'.Auth::user()->profile_photo_path)
        : asset('public/img/logo.png');
}

public function adminlte_desc()
{
    return Auth::user()->cargo;
}
```

FIG 2.13. Ejemplo de Singleton. [Tomado de: Elaboración propia]

Facades

Las facades en Laravel proporcionan una interfaz estática para clases que están en el contenedor de servicios. Muchas de estas clases son Singletons. Por ejemplo, el facade DB para el acceso a la base de datos.

```php
public function backupTable(Request $request){

    $tablas = $request->tablas;

    foreach ($tablas as $tabla) {
        $backup = DB::table($tabla)->get();
        $filename ='['.date('d-m-Y H-i').']'.$tabla.'.sql';

        $datos = json_decode($backup, true);
        $instruccionesSQL = [];

        foreach ($datos as $fila) {
            $columnas = implode(', ', array_keys($fila));
            $valores = implode("', '", array_values($fila));
            $instruccionesSQL[] = "INSERT INTO $tabla ($columnas) VALUES ('$valores');";
        }

        $sqlCompleto = implode("\n", $instruccionesSQL);
        file_put_contents(storage_path('app/public/respaldos/'. $filename), $sqlCompleto);
    }
    return 1;
}
```

FIG 2.14. Ejemplo de Facade, uso de DB. [Tomado de: Elaboración propia.]

Adapter: Permite que dos interfaces incompatibles trabajen juntas. En Laravel, el uso del patrón Adapter puede no ser tan explícito como en otros patrones de diseño, pero se puede encontrar en varios contextos, especialmente cuando Laravel proporciona una capa de abstracción sobre bibliotecas o servicios de terceros para adaptarlos a su propio ecosistema. Es una técnica muy utilizada en el desarrollo de software para adaptar una interfaz de una clase a otra interfaz que se espera utilizar. Esto permite que las clases que no sean compatibles entre sí puedan trabajar juntas y ofrecer una funcionalidad similar. (Leiva, 2022)

```
if ($request->hasFile('foto')) {
    $ruta_guardar = '/';
    $extension_image = trim($request->file('foto')->getClientOriginalExtension());
    $image_subida = Config::get('filesystems.disk.uploads.root');
    $nombre_final = Str::slug($request->name).".".$extension_image;
    $request->foto->storeAs($ruta_guardar, $nombre_final, 'usuario');
    $d->profile_photo_path = $nombre_final;
}
```

FIG 2.15. Uso del patrón Adapter en la Clase UsuarioCrontoller. Elaboración propia.

II.4.3 Tarjetas CRC

A continuación, se describen las tarjetas definidas para la implementación de la solución.

Tabla 2.8. Tarjeta CRC #1. Servicentro.

Tarjeta CRC	
Clase: NServicentroController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista servicentros. • listar(): Obtiene todos los datos de los servicentros. • agregar(Request $request): Adiciona un servicentro. • cargarDatos(Request $request): Obtiene los datos del servicentro. • actualizar(Request $request): Actualiza el servicentro. • eliminar(Request $request): Elimina un servicentro.	Controllers Request NMunicipio NTipoCombustible NServicentro NrSerCombController Rule Validator DB

Tabla 2.9. Tarjeta CRC #2. Distribución.

Tarjeta CRC	
Clase: DistribucionController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista distribucion. • cargarDatosDistribuir(Request $request): Obtiene la información del servicentro. • cargarDatosEntidad(Request $request): Obtiene la información de la entidad. • agregar(Request $request): Agrega una distribución. • listar(Request $request): Obtiene todas las distribuciones. • eliminar(Request $request): Elimina una distribución.	Controllers Request Existencia NEntidad EntidadSaldoTarjeta EntidadDemanda Distribucion NEntidadSub DB Auth

Tabla 2.10. Tarjeta CRC #3. Existencia.

Tarjeta CRC	
Clase: ExistenciaController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista existencia. • agregar(Request $request): Adiciona una nueva entrada y actualiza la existencia.	Controllers Request Existencia NrSerComb Entradas DB

II.5. Conclusiones del capítulo

Terminado este capítulo se arribó a las siguientes conclusiones:

- Luego de definir las historias de usuario se identificaron como principales características funcionales de la solución a desarrollar la creación de roles y permisos para los usuarios, las funciones de crear, consultar, modificar y eliminar en los diferentes módulos, el envío de notificaciones a nivel de sistema, la generación de reportes en formato PDF y la sincronización de datos como mecanismo de intercambio de información entre la aplicación web.
- La entrega de la versión final del sistema a partir de la estimación del tiempo para la implementación de las historias de usuario quedó definida en dos meses y 18 días.

- La arquitectura general del sistema se definió como cliente-servidor utilizando para la aplicación web el patrón de arquitectura Modelo-Vista-Controlador. Se definieron como patrones de diseño GRASP y GoF.
- Se generaron los artefactos correspondientes a las fases de Planificación y Diseño de la metodología de desarrollo XP.

De esta forma quedan definidos los aspectos de arquitectura y diseño a tener en cuenta durante la fase de desarrollo de la solución.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se detallan las iteraciones realizadas durante la etapa de construcción de los módulos propuestos, además se exponen las tareas de ingeniería generadas para cada HU que fueron definidas, así como las pruebas de aceptación planificadas para el sistema. De esta forma es obtenido un producto funcional probado y listo para entregar al cliente al final de cada iteración como propone XP.

III.1. Fase III: Desarrollo

En esta fase, XP plantea que las HU seleccionadas para ser implementadas se realizan durante el transcurso de la iteración a la que pertenecen. Por estas razones, se lleva a cabo una revisión del plan de iteraciones y se modifican en caso de ser necesario. Como parte de este plan se descomponen las HU en tareas de ingeniería (Joskowicz, 2014).

III.2. Tareas de ingeniería

A continuación, se definen las tareas de ingeniería para cada historia de usuario agrupadas según la iteración a la que pertenecen:

Tareas de ingeniería para la Iteración I

Para la primera iteración se definieron un total de 18 tareas de ingeniería con el objetivo de dar cumplimiento a las historias de usuario a la que pertenecen.

Tabla 3.1. Tarea de ingeniería # 1. Implementar el registro y autenticación de usuarios.

Tarea	
Número de tarea: 1	**Número de HU:** 1
Nombre de la tarea: Implementar el registro y autenticación de usuarios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 23 de mayo de 2024	**Fecha de fin:** 24 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa Laravel Jetstream proporciona la implementación para el inicio de sesión, el registro, la verificación de correo electrónico, la autenticación de dos factores, la gestión de sesiones.	

Tabla 3.2. Tarea de ingeniería # 2. Implementar vista de administrador para gestionar los usuarios.

Tarea	
Número de tarea: 2	**Número de HU:** 1
Nombre de la tarea: Implementar vista de administrador para gestionar los usuarios.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 24 de mayo de 2024	**Fecha de fin:** 25 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa una vista para que el administrador pueda tomar decisiones sobre los usuarios, bloquear, editar o eliminar.	

Tabla 3.3. Tarea de ingeniería # 3. Implementar vista del perfil del Usuario.

Tarea	
Número de tarea: 3	**Número de HU:** 1
Nombre de la tarea: Implementar vista del perfil del Usuario.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 27 de mayo de 2024	**Fecha de fin:** 25 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa una vista para que el usuario pueda modificar sus datos, así como actualizar contraseña, número de teléfono, contraseña y demás datos.	

Tabla 3.4. Tarea de ingeniería # 11. Implementar nomenclador de tipo de combustible.

Tarea	
Número de tarea: 11	**Número de HU:** 7
Nombre de la tarea: Implementar nomenclador de tipo de combustible.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 30 de junio de 2024	**Fecha de fin:** 31 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se utiliza el comando *"`php artisan make:model NTipoCombustible -mc`"* la facilidad es agregar un Modelo, Controlador y Migración que se relación entre sí, implementando una vista que permita adicionar, modificar, buscar, actualizar y eliminar.	

Tareas de ingeniería para la Iteración II

Para la segunda iteración se definieron un total de 6 tareas de ingeniería con el objetivo de dar cumplimiento a las historias de usuario a la que pertenecen.

Tabla 3.5. Tarea de ingeniería # 23. Implementar vista del panel informativo.

Tarea	
Número de tarea: 23	**Número de HU:** 15
Nombre de la tarea: Implementar vista del panel informativo.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 17 de julio de 2024	**Fecha de fin:** 18 de julio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementará una vista en la cual se mostrar gráficas que el usuario selecciono como datos importantes, permitiendo su exportación.	

Tabla 3.6. Tarea de ingeniería # 24. Implementar botones de reporte.

Tarea	
Número de tarea: 24	**Número de HU:** 15
Nombre de la tarea: Implementar botones de reporte.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 18 de julio de 2024	**Fecha de fin:** 20 de julio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se agregan botones a las vistas que el usuario seleccione que necesita un reporte específico de ese módulo y se configuran en **`ReporteController`** para de esa forma poder exportarlo en formato PDF.	

Tareas de ingeniería para la Iteración III

Para la segunda iteración se definieron un total de 5 tareas de ingeniería con el objetivo de dar cumplimiento a las historias de usuario a la que pertenecen.

Tabla 3.7. Tarea de ingeniería # 27. Implementar vista de trazas.

Tarea	
Número de tarea: 27	**Número de HU:** 18
Nombre de la tarea: Implementar vista de trazas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 1 de agosto de 2024	**Fecha de fin:** 2 de agosto de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa una vista para que el administrador pueda visualizar los cambios del sistema, estos datos no permiten modificación ni eliminación.	

III.3. Fase IV: Pruebas

III.3.1. Pruebas de aceptación

Las pruebas de aceptación son especificadas por el cliente, se centran en las características y funcionalidades generales del sistema que son visibles. Estas pruebas derivan de las historias de usuario que se han implementado como parte de la liberación del software. Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección y toma de decisiones acerca de estas pruebas. A continuación, se muestra una representación de las pruebas de aceptación a realizarse en cada iteración.

Pruebas de aceptación para la Iteración I

Para la primera iteración, se definieron un total de 20 casos de pruebas de aceptación. Se describe una de las pruebas realizadas.

Tabla 3.8. Prueba de aceptación # 1. Crear usuario desde la vista de registro.

Caso de prueba de aceptación	
Código: HU1_P1	Historia de Usuario: 1
Nombre: Crear usuario desde la vista de registro.	
Descripción: Prueba para la funcionalidad crear usuario desde la vista de registro.	
Condiciones de ejecución: El usuario no puede estar previamente autenticado.	
Pasos de ejecución: 1. El usuario entra a la vista para el registro. 2. El usuario introduce la información solicitada en todos los campos del formulario. 3. Se redirecciona a la vista de login con el mensaje siguiente ***"Su usuario no está activado".*** 4. Los roles de Administrador y Distribuidor reciben una alerta que permite conocer sobre la nueva solicitud y ellos serán los encargados de aprobar o no el acceso. 5. Si el usuario es habilitado se contacta vía SMS y notifica que es posible acceder al sistema.	
Resultado: Satisfactorio	

Pruebas de aceptación para la Iteración II

Para la segunda iteración, se definieron un total de cinco casos de pruebas de aceptación. Se describe una de las pruebas realizadas.

Tabla 3.9. Prueba de aceptación # 23. Adicionar demanda.

Caso de prueba de aceptación	
Código: HU13_P29	Historia de Usuario: 13
Nombre: Adicionar demanda.	
Descripción: Prueba para la funcionalidad adicionar demanda.	
Condiciones de ejecución: El usuario debe estar previamente autenticado en el sistema.	
Pasos de ejecución: 1. El usuario entra a la vista demandas. 2. El usuario presiona el botón  y se abre un modal para la inserción de los datos. 3. El usuario introduce la información solicitada en los campos del formulario que el necesite realizar la demanda. 4. Al finalizar presiona el botón ***Agregar Demanda***.  5. Se le notifica al usuario que se la demanda fue realizada con un mensaje en la esquina superior derecha. 6. Los usuarios del rol de Distribuidor reciben una alerta que permite conocer sobre la nueva solicitud.	
Resultado: Satisfactorio	

Pruebas de aceptación para la Iteración III

Para la tercera iteración, se definieron un total de dos casos de pruebas de aceptación. Se describen una de las pruebas realizadas.

Tabla 3.10. Prueba de aceptación # 26. Crear nueva notificación de usuario creado.

Caso de prueba de aceptación	
Código: HU17_P35	Historia de Usuario: 17
Nombre: Crear nueva notificación de usuario creado.	
Descripción: Prueba para la funcionalidad crear nueva notificación de usuario creado.	
Condiciones de ejecución: El usuario no puede estar previamente autenticado en el sistema y se debe haber realizado el registro de usuario.	
Pasos de ejecución: 1. El usuario completa los campos del registro de usuario y presiona el botón ***Registrarse***. 2. Los usuarios del rol Administrador y Distribuidor reciben una alerta que permite conocer sobre la nueva solicitud.	
Resultado: Satisfactorio	

Pruebas de aceptación para la Iteración IIII

Para la cuarta iteración, no se definieron casos de prueba, lo que permitió comprobar el estado de aceptación del sistema.

III.3.2. Análisis de las pruebas de aceptación

Se aplicaron un total de 27 pruebas de aceptación distribuidas en las diferentes iteraciones que se realizaron. A continuación, se muestran en gráficas los resultados de satisfacción alcanzados en cada iteración, ver FIG 3.1

Fig 3.1. Pruebas de aceptación distribuidas por iteraciones. Fuente: Elaboración propia.

Como se puede observar en la representación, en la primera iteración se realizaron 20 pruebas, de ellas 15 obtuvieron un nivel de satisfacción esperado, mientras que 5 de las pruebas se detectaron errores en la validación de entrada de datos en los formularios. En tanto, en la segunda iteración se realizaron cinco pruebas, de ellas tres fueron satisfactorias. En la tercera iteración dos pruebas, obteniendo las dos satisfactorias y la cuarta iteración 0 pruebas lo que demostró la satisfacción del sistema. Con estas comprobaciones se evidencia una correcta implementación de las funcionalidades trazadas.

III.3.3. Prueba de Seguridad.

En el desarrollo de software la seguridad es un requisito indispensable para garantizar la calidad del producto que se desarrolla. Las pruebas de seguridad se realizan con el objetivo de encontrar vulnerabilidades en los sistemas de cómputo para posteriormente mitigarlas en la medida de lo posible. Si bien es casi imposible garantizar este aspecto de calidad al 100%,

si se pueden realizar una serie de pruebas que permitan conocer donde se encuentran las debilidades y corregirlas (SOMMERVILLE, 2005). Se configuro el entorno de trabajo con la aplicación Laragon 6.0 que permite ejecutar los servicios necesarios para visualizar la web. Para probar la realidad de la web para la asignación de combustibles se utilizó la herramienta Acunetix obteniendo los siguientes resultados.

Iteración No.1

Fueron detectadas un total de 3 alertas de información y 12 vulnerabilidades donde las más altas de seguridad se identificaron por acceso a directorios y el archivo .env donde se alojan configuraciones críticas de la aplicación.

Alerts distribution

Total alerts found	15
High	3
Medium	2
Low	7
Informational	3

FIG 3.2 Vulnerabilidades detectadas en la iteración número 1. Elaboración Propia.

Iteración No.2

Fueron detectadas un total de 1 alerta de información y de 4 vulnerabilidades quedando por resolver una de complejidad media, siendo esta, Ataque de Denegación de Servicio HTTP Lento.

Alerts distribution

Total alerts found	5
High	0
Medium	1
Low	3
Informational	1

FIG 3.3 Vulnerabilidades detectadas en la iteración número 2. Elaboración Propia.

Iteración No.3

Fueron detectadas un total de 1 alerta de información y de 0 vulnerabilidad asociadas a la información que devuelve el servidor apache sobre la versión PHP.

Alerts distribution

Total alerts found	1
High	0
Medium	0
Low	0
Informational	1

FIG 3.4 Vulnerabilidades detectadas en la iteración número 3. Elaboración Propia.

III.3.4. Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Las pruebas deben ser definidas antes de realizar el código (López, 2016).

Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo (López, 2016).

En este proyecto, se utilizó el marco de pruebas de Laravel, que se basa en PHPUnit, para implementar pruebas unitarias que aseguran la calidad y robustez de la aplicación desarrollada.

III.3.4.1 Herramientas y Entorno

PHPUnit: Es un marco de pruebas para PHP que permite realizar pruebas unitarias de manera sencilla y estructurada. Es ampliamente utilizado en la comunidad de PHP y está integrado con Laravel, permitiendo ejecutar pruebas y obtener reportes de manera eficiente.

El entorno de pruebas se configuró utilizando Laravel, que proporciona un conjunto completo de herramientas para realizar pruebas de manera eficiente. Se utilizó el comando **php artisan test** para ejecutar las pruebas y garantizar que cada componente funcionara según lo esperado.

Configuración del Entorno de Pruebas:

Laravel ofrece un archivo de configuración específico para pruebas llamado .env.testing, que permite establecer variables de entorno que solo se utilizarán durante la ejecución de pruebas. Esto es útil para evitar que las pruebas interactúen con la base de datos de producción o con otros servicios críticos.

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="./vendor/phpunit/phpunit/phpunit.xsd"
         bootstrap="vendor/autoload.php"
         colors="true"
>
    <testsuites>
        <testsuite name="Unit">
            <directory suffix="Test.php">./tests/Unit</directory>
        </testsuite>
        <testsuite name="Feature">
            <directory suffix="Test.php">./tests/Feature</directory>
        </testsuite>
    </testsuites>
    <coverage processUncoveredFiles="true">
        <include>
            <directory suffix=".php">./app</directory>
        </include>
    </coverage>
    <php>
        <server name="APP_ENV" value="testing"/>
        <server name="BCRYPT_ROUNDS" value="4"/>
        <server name="CACHE_DRIVER" value="array"/>
        <!-- <server name="DB_CONNECTION" value="sqlite"/> -->
        <!-- <server name="DB_DATABASE" value=":memory:"/> -->
        <server name="MAIL_MAILER" value="array"/>
        <server name="QUEUE_CONNECTION" value="sync"/>
        <server name="SESSION_DRIVER" value="array"/>
        <server name="TELESCOPE_ENABLED" value="false"/>
    </php>
</phpunit>
```

FIG 3.5 Fichero de configuración de variables de pruebas. Elaboración Propia.

III.3.4.2. Desarrollo Guiado por Pruebas (TDD)

1. **Definición**:
    - El Desarrollo Guiado por Pruebas (TDD, por sus siglas en inglés) es una metodología que prioriza la escritura de pruebas antes de desarrollar la funcionalidad del código. Esto implica que antes de escribir el código que implementa una característica, primero se escribe una prueba que describe el comportamiento esperado de dicha característica.

2. **Ciclo de TDD**:

    - **Rojo**: Se escribe una prueba que inicialmente falla porque la funcionalidad no está implementada. Esto garantiza que la prueba es válida.
    - **Verde**: Se implementa la mínima cantidad de código necesario para que la prueba pase.
    - **Refactorizar**: Se optimiza el código, asegurando que las pruebas sigan pasando. Este paso es crucial para mantener la calidad del código y facilitar futuros cambios.

III.3.4.3. Estructura de las Pruebas

1. **Estructura Dado-Cuando-Entonces (Given-When-Then)**:
    - Esta estructura se utiliza para organizar las pruebas de manera clara y comprensible.
    - **Dado (Given)**: Describe el estado inicial del sistema antes de que se realice la acción que se está probando.
    - **Cuando (When)**: Indica la acción o evento que desencadena la prueba.
    - **Entonces (Then)**: Define el resultado esperado después de que se haya realizado la acción.

2. **Ejemplo de la Estructura**:
    - **Dado**: Dado que un usuario se ha registrado correctamente en la aplicación.
    - **Cuando**: Cuando el usuario intenta iniciar sesión con sus credenciales.

- **Entonces**: Entonces se debe permitir el acceso al sistema y redirigir al usuario a su panel de control.

III.3.4.4 Ejecución de Pruebas:

A continuación, se presentan algunos ejemplos de pruebas unitarias realizadas durante el desarrollo de la aplicación:

```php
public function testRegistrarUsuario()
{
    /** @test */
    $data = [
        'name' => 'Juan Pérez',
        'email' => 'sdf27@example.com',
        'password' => 'password',
        'telefono' => '58490447',
        'id_entidad' => '1',
        'cargo' => 'Cargo de Prueba',
    ];

    $usuario = User::create($data);

    $this->assertInstanceOf(User::class, $usuario);
    $this->assertEquals('sdf27@example.com', $usuario->email);
}
```

FIG 3.6 Prueba Registrar Usuario. Elaboración Propia.

```php
public function testRegistrarCategoria()
{
    /** @test */
    $datos = [
        'nombre' => 'Categoria de prueba',
    ];

    $categoria = NCategoria::create($datos);

    $this->assertInstanceOf(NCategoria::class, $categoria);
    $this->assertEquals('Categoria de prueba', $categoria->nombre);
}
```

FIG 3.7 Prueba Agregar Categoría. Elaboración Propia.

```php
public function testRegistrarEntidad()
{
    /** @test */
    $datos = [
        'nombre' => 'Entidad 55',
        'id_categoria' => '1',
        'id_municipio' => '1',
    ];

    $entidad = NEntidad::create($datos);

    $this->assertInstanceOf(NEntidad::class, $entidad);
    $this->assertEquals('Entidad 55', $entidad->nombre);
}
```

FIG 3.8 Prueba Agregar Entidad. Elaboración Propia.

Las pruebas se pueden ejecutar utilizando el comando php artisan test desde la terminal. Este comando ejecuta todas las pruebas en el directorio tests y proporciona un resumen de los resultados en la consola.

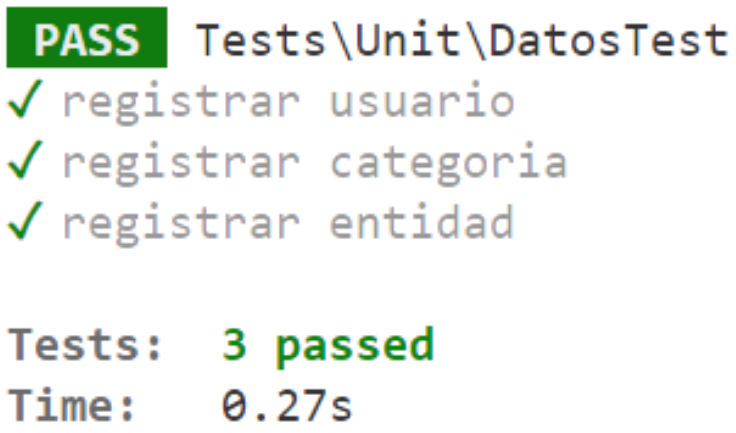


FIG 3.9 Resultado de pruebas realizadas. Elaboración Propia.

Se realizaron pruebas al final de cada iteración contando al concluir con un total de 12 pruebas. El gráfico que se muestra a continuación visualiza los resultados obtenidos en el proceso de pruebas unitarias.

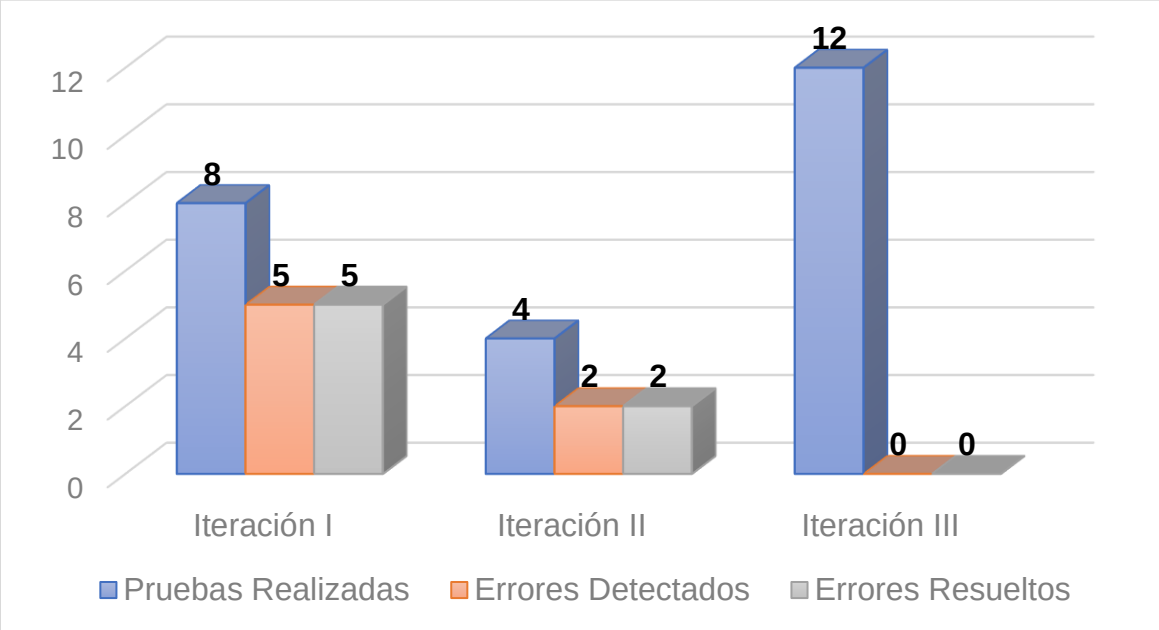


FIG 3.10 Resultado de pruebas unitarias por iteración. Elaboración Propia.

III.3.5. Prueba de Rendimiento

La prueba de rendimiento se diseña para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado de rendimiento de software. Se realizan para medir la respuesta de la aplicación a distintos volúmenes de carga esperados, se centra en determinar la velocidad con la que el sistema bajo pruebas realiza una tarea en las condiciones particulares del escenario de pruebas (Pressman R. S., 2010).

Resultados de las pruebas de rendimiento

Para las pruebas de rendimiento se utiliza el software Apache Jmeter v5.1.1. Se configuraron los parámetros del Apache JMeter logrando un ambiente de simulación con un total de 50 usuarios conectados concurrentemente. En la Figura 3.10 y Fig 3.11 se puede observar los resultados obtenidos por el sistema. Para un mejor entendimiento de los componentes que se verán a continuación, se explica cada parámetro que la compone:

Muestras: cantidad de hilos utilizados para la URL.

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

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

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

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

Kb/sec: rendimiento medido en Kbyte por segundo.

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Listar Entidades	100	1323	367	3239	655,24	0,00%	4,7/sec	139,05	0,64	30088,8
Total	100	1323	367	3239	655,24	0,00%	4,7/sec	139,05	0,64	30088,8

FIG 3.11 Resultado de carga a Listar Entidades

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Listar Distribuciones	100	1291	350	3402	644,49	0,00%	2,0/sec	1,10	0,28	556,0
Total	100	1291	350	3402	644,49	0,00%	2,0/sec	1,10	0,28	556,0

FIG 3.13 Resultado de carga a Listar Distribuciones

Análisis de los resultados de las pruebas de rendimiento

El tiempo promedio de las solicitudes para la primera petición es de 1323 milisegundos, realizándose un total de 100 solicitudes al servidor. El tiempo promedio de las solicitudes para la segunda petición es de 1291 milisegundos, realizándose un total de 100 solicitudes al servidor. Ambas pruebas demostraron que el sistema puede trabajar con más de 40 peticiones simultaneas cumpliendo así el **RNF22**.

III.4. Conclusiones del capítulo

Como conclusiones en este capítulo:

- Se especificó el proceso de implementación del sistema a partir del desglose de las HU en tareas de ingeniería, lo que permitió especificar los procedimientos necesarios para dar cumplimiento a cada HU.
- Se definieron y aplicaron las pruebas de aceptación a las funcionalidades de los módulos desarrollados. Estas pruebas detectaron siete fallas en el sistema lo que posibilitó corregirlas y mejorar la operabilidad del mismo.

- Al aplicar las pruebas de seguridad en las tres iteraciones planificadas se encontraron 15, tres alertas de información y doce vulnerabilidades, en las cuales se trabajaron y en la tercera iteración solo quedo pendiente 1 alerta de información.
- Al realizar las pruebas unitarias al sistema propuesto se identificaron en las doce pruebas realizadas siete errores, permitiendo su corrección y haciendo más eficiente el sistema.
- Al utilizar JMeter para la carga y estrés del sistema se pudo comprobar que el sistema soporta hasta 60 usuarios conectados simultáneamente realizando 100 peticiones en un minuto sin colapsar el sistema.
- Con las pruebas antes mencionadas se llega a la conclusión de que el sistema de gestión cumple con las exigencias del cliente.

CONCLUSIONES FINALES

Con el desarrollo de la presente investigación se arriba a las siguientes conclusiones:

- Se logró establecer los fundamentos y referentes teórico-metodológicos para el desarrollo de un sistema informático para la gestión de combustible quedando evidenciado la carencia de un sistema de este tipo en la provincia, así como la importancia de contar con uno propio.
- La utilización de las tecnologías y herramientas definidas permitió que la solución informática desarrollada sea un referente para el país, obteniendo un sistema que mejorará la asignación del combustible, brindando un mejor control y racionalización de tan importantes combustibles.
- Al realizar el análisis se obtuvieron 21 historias de usuarios donde se definieron todas las necesidades que presentaba el cliente, realizando por parte del desarrollador 3 iteraciones entregables permitiendo la corrección de errores en su utilización.
- Las pruebas de aceptación, seguridad, unitarias y de rendimiento realizadas al software implementado demostraron el cumplimiento de los atributos de calidad, lo que se demuestra en el resultado obtenido por las mismas en la evaluación de los módulos del sistema, los cuales cumple satisfactoriamente con los requisitos definidos por los clientes.
- Se implementó la web de asignación de combustible en la sede del gobierno provincial del poder popular de Artemisa con éxito.

RECOMENDACIONES

A partir de los resultados obtenidos se considera que son relevantes e incrementarían la utilidad del sistema y la calidad de uso del mismo por lo que se recomienda:

- Desarrollar una aplicación móvil que permita a los usuarios una interacción más directa con los permisos básicos de demandar y consultar asignaciones.

REFERENCIAS

Díaz, D., & Jesús, A. (2024). *El combate contra el delito y las ilegalidades, tarea de todos.* Recuperado el 25 de 06 de 24, de https://www.presidencia.gob.cu/es/noticias/el-combate-contra-el-delito-y-las-ilegalidades-tarea-de-todos/

Asamblea Nacional del Poder Popular. (2024). *Órganos Locales del Poder Popular.* Recuperado el 03 de 09 de 2024, de https://www.parlamentocubano.gob.cu/organos-locales-del-poder-popular

Bernabeu, D. (2021). *Por qué Apache ECharts es la mejor librería gráfica JavaScript?* Recuperado el 25 de 06 de 24, de https://es.linkedin.com/pulse/por-que-apache-echarts-es-la-mejor-libreria-grafica-dario-bernabeu#:~:text=Apache%20ECharts%20es%20una%20librer%C3%ADa%20JavaScript%20Open%20Source,un%20entorno%20web%2C%20liviano%20y%20de%20r%C3%A1pida%20carga.

Bertalanffy, K. L. (1928). *La Teoría General de Sistemas, de Ludwig von Bertalanffy.* Recuperado el 25 de 06 de 24, de https://psicologiaymente.com/psicologia/teoria-general-de-sistemas-ludwig-von-bertalanffy

Bourque, P. &. (2004). *Guide to the Software Engineering Body of Knowledge (SWEBOK).* IEEE Computer Society Press.

CASAS, V. (2019). *¿Qué es Bootstrap y para qué sirve?* Obtenido de https://www.lucushost.com/blog/que-es-bootstrap/

Chávez, G. (2024). *¿Qué es Laravel Jetstream?* Recuperado el 25 de 06 de 24, de https://gabrielchavez.me/que-es-laravel-jetstream/

CloudFleet. (2024). *CloudFleet.* Recuperado el 11 de 09 de 2024, de https://cloudfleet.com/features

Cursos de programación Online. (2022). *AdminLTE: Plantilla de administración para Bootstrap.* Recuperado el 04 de 07 de 2024, de https://www.cursosdesarrolloweb.es/blog/adminlte-plantilla-administracion-bootstrap

Delgado. (2021). *¿Qué es un IDE (entorno de desarrollo integrado)?* Obtenido de https://ourcodeworld.co/articulos/leer/1469/que-es-un-ide-entorno-de-desarrollo-integrado

Deqing Li, H. M. (2018). *ECharts: A declarative framework for rapid construction of web-based visualization.* Obtenido de https://www.sciencedirect.com/science/article/pii/S2468502X18300068

EChartsTM, A. (s.f.). *Features.* Recuperado el 25 de 06 de 24, de https://echarts.apache.org/en/feature.html

Edraw. (2024). *Ejemplos de diagramas UML.* Obtenido de https://www.edrawsoft.com/es/uml-diagrama-examples.html

Endeos, B. (2024). *Notificaciones al usuario con jQuery Sweet Alert.* Recuperado el 25 de 06 de 24, de https://blog.endeos.com/notificaciones-al-usuario-con-jquery-sweet-alert/

EUGCOM. (2009). *Descripción del Producto ADMINISTRACIÓN Y CONTROL DE FLOTAS.*

Figueredo Reinaldo, O., Sifonte Díaz, Y. J., Doimeadios Guerrero, D., Romeo Matos, L., Carmona Tamayo, E., & Padrón Padilla, A. (2019). *Robo de combustible: Fugas y coartadas (+ Infografías).* Recuperado el 25 de 06 de 24, de http://www.cubadebate.cu/especiales/2019/09/04/robo-de-combustible-fugas-y-coartadas-infografias/

Flanagan, D. (2020). *JavaScript: The Definitive Guide, 7th Edition.* Obtenido de https://www.oreilly.com/library/view/javascript-the-definitive/9781491952016/

Gaceta Oficial. (2020). *De Organización y Funcionamiento del Gobierno Provincial del Poder Popular.* Recuperado el 25 de 06 de 24, de https://www.gacetaoficial.gob.cu/es/ley-138-de-2020-de-asamblea-nacional-del-poder-popular

Gaceta Oficial. (2022). *De las estructuras organizativas en las administraciones provinciales del Poder Popular.* Recuperado el 25 de 06 de 24, de https://www.gacetaoficial.gob.cu/es/decreto-69-de-2022-de-consejo-de-ministros

Geek. (2020). *Qué es un Entorno de desarrollo integrado.* Obtenido de https://ifgeekthen.nttdata.com/es/que-es-un-entorno-de-desarrollo-integrado

Gonzalez., D. R. (2024). *Concepto de Combustible: Ejemplos, para que sirve y según autores.* Recuperado el 25 de 06 de 24, de https://conceptopedia.de/combustible-ejemplos-para-que-sirve-segun-autores/

Gonzalez., D. R. (2024). *Concepto de Gestión: Según Autores y Definición.* Recuperado el 25 de 06 de 24, de https://conceptopedia.de/gestion-segun-autores-definicion/

Guamán, V. (2021). *¿Qué es MVC? Lo que deberías saber acerca de este patrón de arquitectura de software.* Recuperado el 09 de 07 de 2024, de https://dev.to/veronicaguamann/que-es-mvc-lo-que-deberias-saber-acerca-de-este-patron-de-arquitectura-de-software-5hhe

Hostinger. (2024). *¿Qué es un hosting y cómo funciona?* Recuperado el 25 de 06 de 24, de https://www.hostinger.mx/tutoriales/que-es-un-hosting#%C2%BFQue_es_un_hosting

IncanatoIt. (s.f.). *Diseño Sistema web PHP con laravel y Mysql (4-36) Rutas y Modelo en laravel .* Obtenido de https://www.incanatoit.com/2016/07/sistema-web-php-laravel-mysql-rutas-modelo.html

Invarato, R. (2019). *Cliente vs Servidor.* Obtenido de https://jarroba.com/cliente-vs-servidor/

Joskowicz, J. (2014). *DESARROLLO DE SOFTWARE UTILIZANDO LA METODOLOGÍA XP.* Recuperado el 10 de 07 de 2024, de https://www.clubensayos.com/Ciencia/DESARROLLO-DE-SOFTWARE-UTILIZANDO-LA-METODOLOG%C3%8DA-XP/1769879.html

Knowdo. (2018). *Sistemas Web.* knowdo.org. Obtenido de http://www.knowdo.org/knowledge/39-sistemas-web

Lawson, B., & Sharp, R. (2011). *Introducing HTML5.* Obtenido de https://books.google.com/books/about/Introducing_HTML5.html?id=a8HQCk4pbQkC

Leiva, A. (05 de 12 de 2022). *Adapter – Patrones de Diseño*. Recuperado el 02 de 06 de 2024, de Devexpert: https://devexpert.io/adapter-patrones-diseno/

López, Y. B. (2016). *Metodología Ágil de Desarrollo de Software – XP.* ESPE, MEVAST.

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software.* Obtenido de https://repositorio.uca.edu.ar/handle/123456789/522

MariaDB Foundation. (2024). *MariaDB: The Open Source Database for Developers.* Recuperado el 25 de 06 de 24, de https://mariadb.org/

Martínez, C. C. (s.f.). *Fundamentos y evolución de la multimedia.* Recuperado el 09 de 07 de 2024, de http://multimedia.uoc.edu/blogs/fem/es/las-aplicaciones-web-y-las-bases-de-datos/

McGregor, M. (21 de 04 de 2023). *4 patrones de diseño que deberías saber para desarrollo web: Observador, Singleton, estrategia, y decorador*. Recuperado el 02 de 06 de 2024, de https://www.freecodecamp.org/espanol/news/4-patrones-de-diseno-que-deberias-saber-para-desarrollo-web-observador-singleton-estrategia-y-decorador/

Muente, G. (2020). *Guía completa del Framework: qué es, cuáles tipos existen y por qué es importante en Internet.* Obtenido de https://rockcontent.com/es/blog/framework/

Naranjo, F. J. (2015). *Sistemas de Gestión: Valor Estratégico de las Organizaciones.*

Olivares Rodríguez, Y., Aparicio Reytor, M., & De Armas Rodríguez, N. (2011). *MÓDULO DE ANÁLISIS DE REDES SOCIALES PARA EL SISTEMA DE VIGILANCIA TECNOLÓGICA DEL GRUPO GESTIÓN DE LA INFORMACIÓN Y EL CONOCIMIENTO.* Obtenido de https://repositorio.uci.cu/jspui/handle/ident/8169

Padinger, G. (2021). *En qué consiste el embargo de EE.UU. a Cuba y cómo ha afectado la economía de la isla.* Obtenido de https://cnnespanol.cnn.com/2021/11/09/embargo-eeuu-cuba-afectado-economia-isla-orix/

Portal del Ciudadano, P. A. (25-6-24). *Gobernador y Vicegobernadora en Artemisa.* Obtenido de https://artemisa.gob.cu/es/consejo-provincial-de-gobierno

Porto, J. P. (2021). *Gestión - Qué es, tipos, definición y concepto.* Definicion.de. Obtenido de https://definicion.de/gestion/

POWER, G. (2023). *¿Cómo crear una tabla con DataTable?*

Pressman, R. S. (2010). *Ingeneria de Software. Un enfoque práctico* (7 ed.). MCGRAW-HILL. Recuperado el 01 de 06 de 2024

Pressman, R. S. (2010). *Ingeniería de software: Un enfoque práctico.* McGraw-Hill.

ProximaHost. (2024). *Laravel, todo lo que debes saber sobre este framework para desarrollo web.* Recuperado el 25 de 06 de 24, de https://proximahost.es/blog/laravel-framework-desarrollo-web/

Raeburn, A. (2024). *La programación extrema (XP) produce resultados, pero ¿es la metodología adecuada para ti?* Obtenido de https://asana.com/es/resources/extreme-programming-xp

Reinaldo, O. F. (2024). *Entran en vigor nuevos precios del combustible y la electricidad a partir del primero de marzo.* Obtenido de http://www.cuba.cu/economia/2024-02-28/entran-en-vigor-nuevos-precios-del-combustible-y-la-electricidad-a-partir-del-primero-de-marzo/64866

Saavedra, A. R. (2013). *Desarrollo del módulo Control de combustible del Sistema de Control de Flota y Mantenimiento de la Dirección de Transporte de la Universidad de las Ciencias Informáticas.* Obtenido de http://bibliodoc.uci.cu/RDigitales/2013/noviembre/26/TD_06861_13.pdf

Solomon, E. (2021). *Novedades de Laravel 9: Una Inmersión Profunda en la Próxima Gran Versión.* Obtenido de https://kinsta.com/es/blog/laravel-9/

SOMMERVILLE, I. y. (2005). *Ingeniería del software.* S.l.: Pearson Educación.

Trends, G. (2024). *Google Trends.* Recuperado el 11 de 09 de 2024, de https://trends.google.es/trends/explore?q=%2Fm%2F0jwy148,%2Fm%2F06y_qx,%2Fm%2F02qgdkj,%2Fm%2F0dgs72v,%2Fm%2F0_v2szx&hl=es

Unir. (2022). *Framework: qué es, para qué sirve y algunos ejemplos.* Obtenido de https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/

Universidades, S. (2020). *Metodologías de desarrollo de software: ¿qué son?* Obtenido de https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html

Vaswani, V. (2010). *Fundamentos de PHP.* Obtenido de https://studylib.es/doc/8856044/fundamentos-de-php

Velasco, R. (2021). *Visual Studio Code: el editor de código de Microsoft que querrás instalar.* Obtenido de https://www.softzone.es/programas/utilidades/visual-studio-code/

Yánez Triana, C. K. (2022). *Estudio comparativo de las herramientas de metodologías ágiles para el aplicar buenas prácticas de desarrollo en la calidad de software.* Obtenido de http://dspace.utb.edu.ec/handle/49000/11852

ANEXOS

Tabla 4: Historia de Usuario #1. Gestionar Usuario.

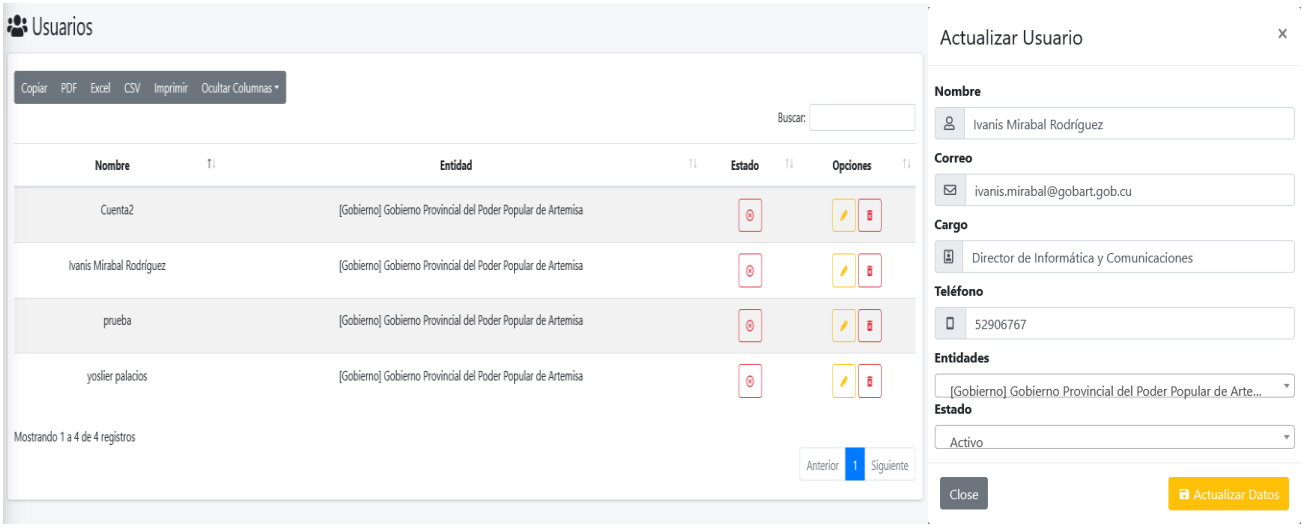
Historias de Usuarios	
Número: HU_1	**Nombre del Requisito**: Gestionar Usuario
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Muy Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de los usuarios es clave en el sistema. Se permite modificar y eliminar datos desde la columna de opciones. • **Mostrar:** Visualiza todos los usuarios con sus columnas y opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Modificar:** Permite cambiar todos los datos previamente ingresados. • **Eliminar:** Permite eliminar el usuario seleccionado.	
Prototipo de Interfaz: 	

Tabla 5: Historia de Usuario #2. Gestionar Rol.

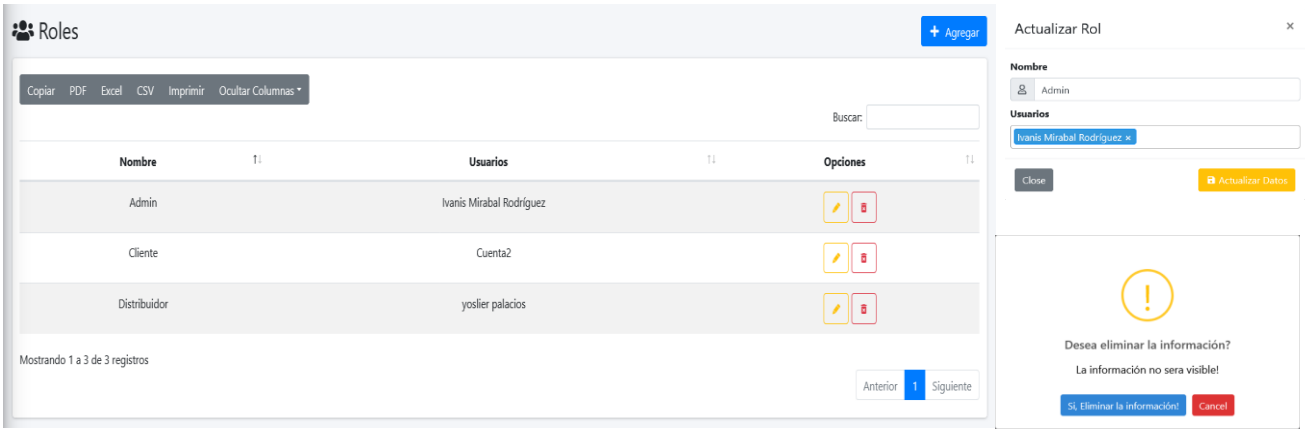
Historias de Usuarios	
Número: HU_2	**Nombre del Requisito**: Gestionar Rol
Rol de Usuario: Administrador	**Tiempo estimado**: 1.0
Prioridad: Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de los roles es elemental en el sistema. Es donde se agruparon los usuarios para otorgar los permisos. • **Mostrar:** Visualiza todos los roles con los usuarios adjudicados y las opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Adicionar:** Permite agregar un nuevo rol al sistema, además de agregar los usuarios a ese rol. • **Modificar:** Permite cambiar todos los datos previamente ingresados, además de modificar los usuarios en ese rol. • **Eliminar:** Permite eliminar el rol seleccionado y con ello se eliminan los permisos y usuarios asociados.	
Prototipo de Interfaz: 	

Tabla 6: Historia de Usuario #3. Gestionar Permisos.

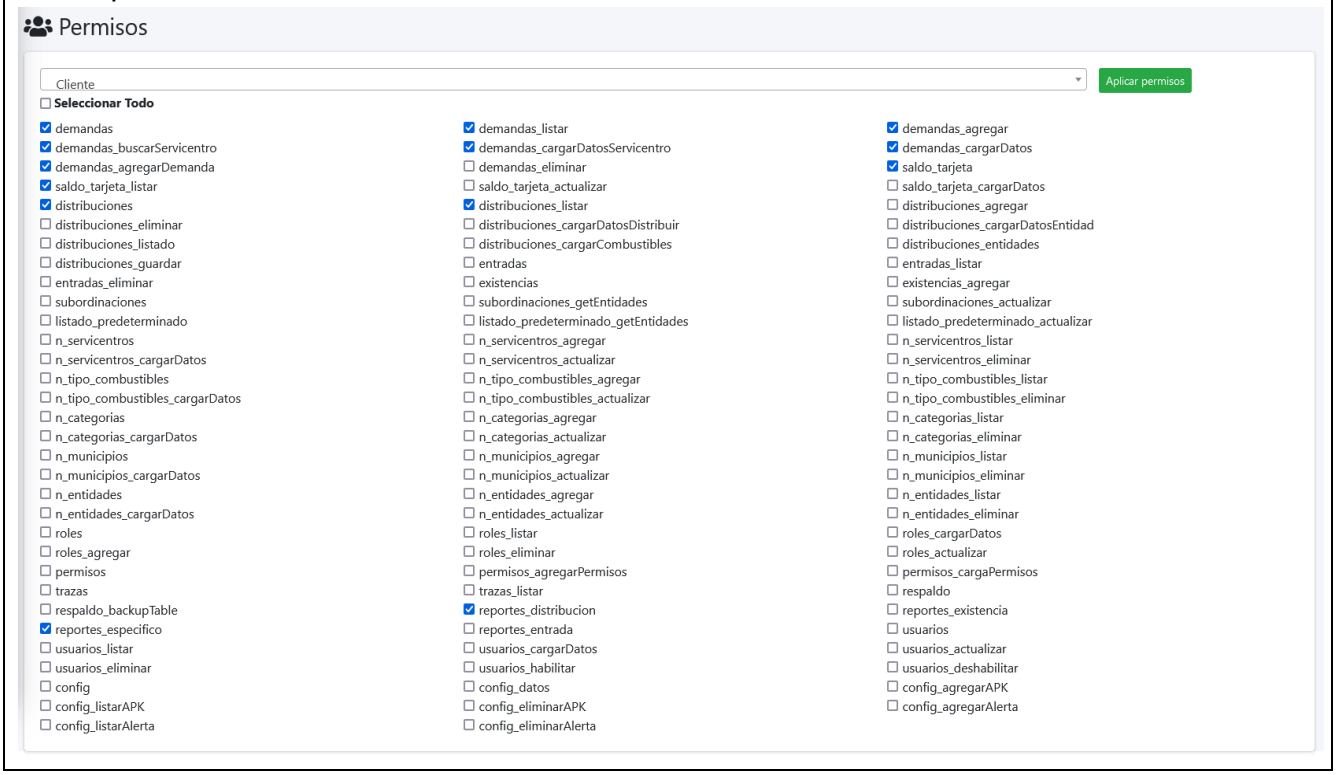
Historias de Usuarios	
Número: HU_3	**Nombre del Requisito**: Gestionar Permisos
Rol de Usuario: Administrador	**Tiempo estimado**: 1.0
Prioridad: Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de los permisos es donde el administrador definirá que responsabilidad ocupará el usuario dentro del sistema. • **Listar:** Muestra un seleccionar en la parte superior que permite seleccionar el rol al cual se le asignaran, modificaran o eliminaran permisos. • **Mostrar:** Muestra todos los permisos disponibles en el sistema. • **Aplicar Permisos:** Una vez seleccionador el rol, se seleccionarán los permisos que ostenta, el administrador actúa sobre ellos y aplica los cambios.	
Prototipo de Interfaz: 	

Tabla 7: Historia de Usuario #4. Gestionar Categoría.

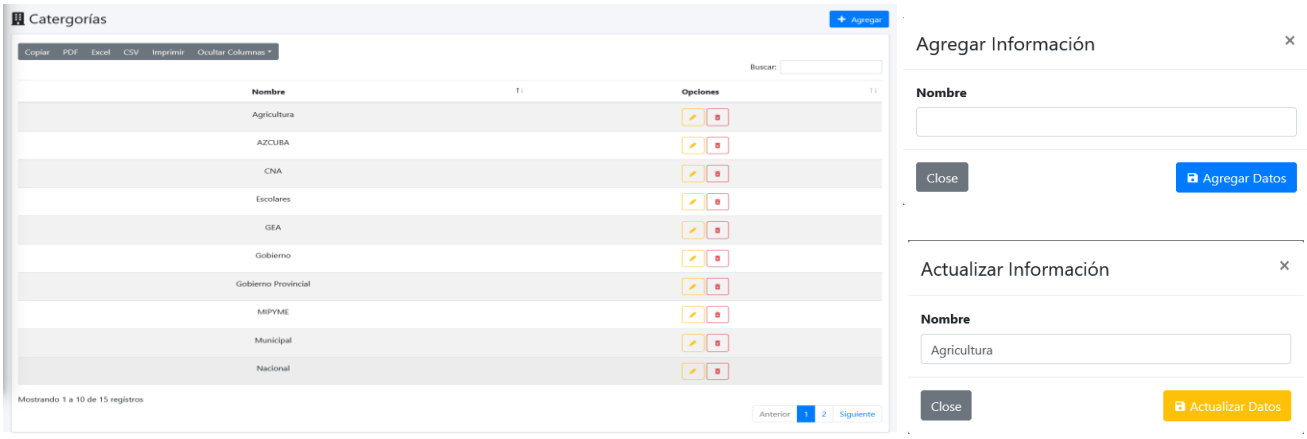
Historias de Usuarios	
Número: HU_4	**Nombre del Requisito**: Gestionar Categoría
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Media	**Riesgo en desarrollo**: Bajo
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de las categorías es donde el usuario, como indica el nombre, creara categorías para agrupar las empresas. • **Mostrar:** Visualiza todos las categorías y las opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Adicionar:** Permite agregar una nueva categoría al sistema. • **Modificar:** Permite cambiar los datos previamente ingresados. • **Eliminar:** Permite eliminar la categoría.	
Prototipo de Interfaz: 	

Tabla 8: Historia de Usuario #5. Gestionar Municipio.

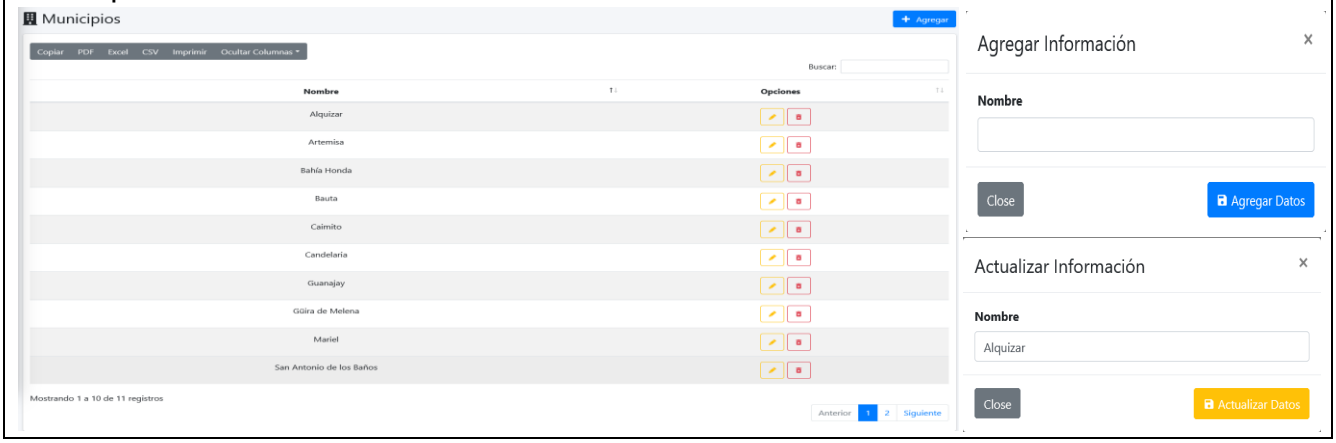
Historias de Usuarios	
Número: HU_5	**Nombre del Requisito**: Gestionar Municipio
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Media	**Riesgo en desarrollo**: Bajo
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de los municipios es donde el usuario, como indica el nombre, creara municipios para agrupar las empresas. • **Mostrar:** Visualiza todos los municipios y las opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Adicionar:** Permite agregar un nuevo municipio al sistema. • **Modificar:** Permite cambiar los datos previamente ingresados. • **Eliminar:** Permite eliminar el municipio.	
Prototipo de Interfaz: 	

Tabla 9: Historia de Usuario #6. Gestionar Entidad.

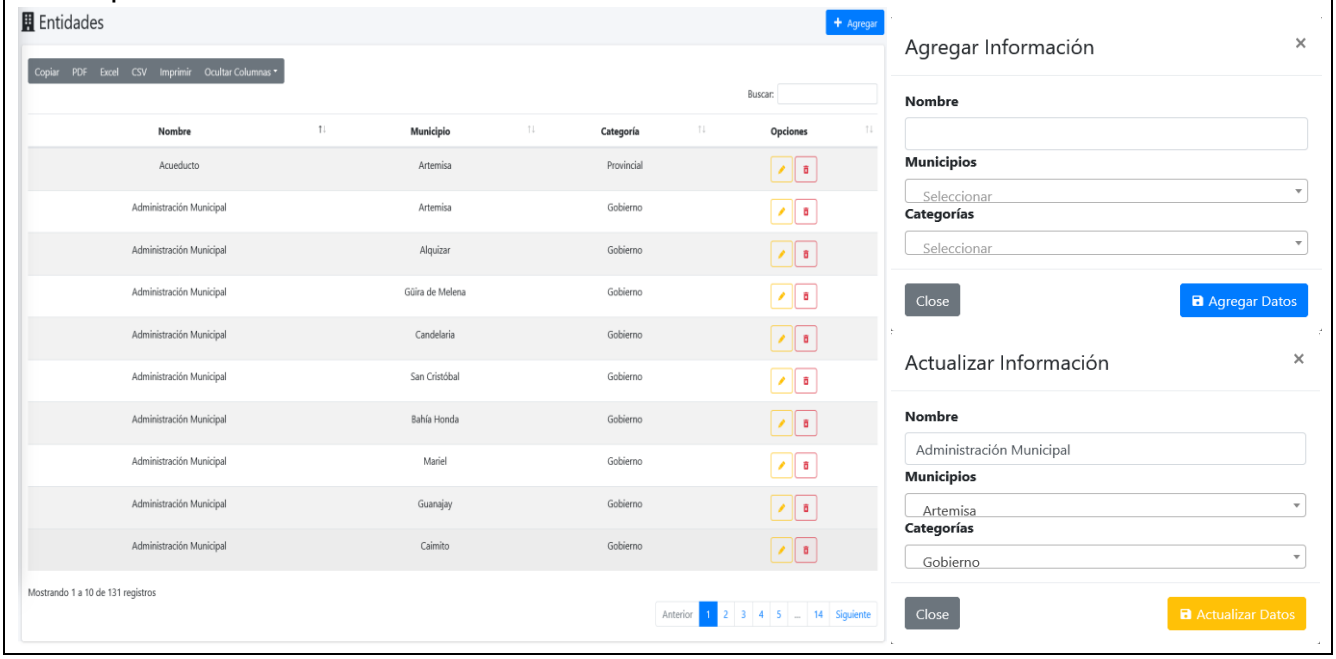
Historias de Usuarios	
Número: HU_6	**Nombre del Requisito**: Gestionar Entidad
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Muy Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de las entidades es donde el usuario creara las entidades y las agrupa por las categorías y municipios. • **Mostrar:** Visualiza todos las entidades y las opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Adicionar:** Permite agregar una nueva entidad al sistema. • **Modificar:** Permite cambiar los datos previamente ingresados. • **Eliminar:** Permite eliminar la entidad.	
Prototipo de Interfaz: 	

Tabla 10: Historia de Usuario #7. Gestionar Tipo de Combustible.

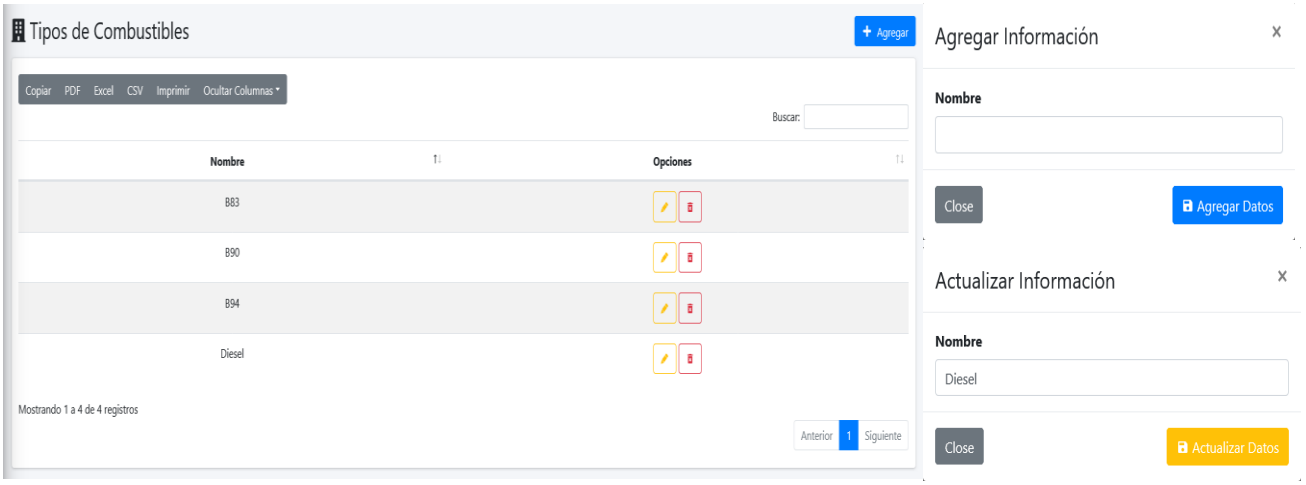
Historias de Usuarios	
Número: HU_7	**Nombre del Requisito**: Gestionar Tipo de Combustible
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Muy Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de los tipos de combustibles es donde se definirán los combustibles asignar. • **Mostrar:** Visualiza todos los tipos de combustibles y las opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Adicionar:** Permite agregar un nuevo tipo de combustible al sistema. • **Modificar:** Permite cambiar los datos previamente ingresados. • **Eliminar:** Permite eliminar el tipo de combustible.	
Prototipo de Interfaz: 	

Tabla 11: Historia de Usuario #10. Gestionar Entrada.

Historias de Usuarios	
Número: HU_10	**Nombre del Requisito**: Gestionar Entrada
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de las entradas se basa en visualizar las entradas de combustibles separadas por fecha, servicentro y tipo de combustible, la única opción permisible es eliminar. • **Mostrar:** Visualiza todos los tipos de combustibles y las opciones. • **Buscar:** Permite buscar dinámicamente los datos en la tabla. • **Eliminar:** Permite eliminar la entrada y de esta forma es restado a la existencia de ese servicentro en ese tipo de combustible.	
Prototipo de Interfaz:	

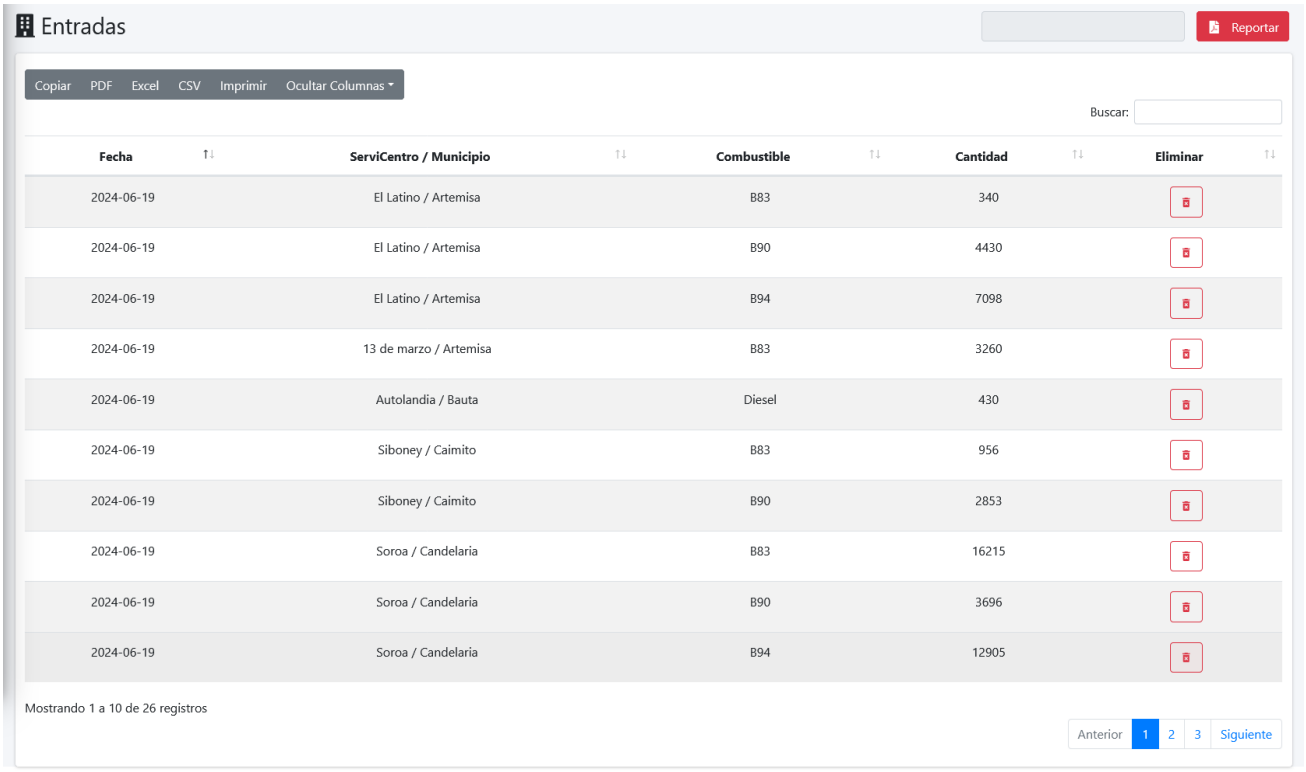


Tabla 12: Historia de Usuario #12. Gestionar Demanda.

Historias de Usuarios	
Número: HU_12	**Nombre del Requisito**: Gestionar Demanda
Rol de Usuario: Cliente, Distribuidor, Administrador	**Tiempo estimado**: 1.0
Prioridad: Media	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	

Descripción: La gestión de las demandas es donde la entidad tendrá acceso para realizar su solicitud por cada tipo de combustible para el día.

- **Mostrar:** Visualiza las demandas de la entidad.
- **Buscar:** Permite buscar dinámicamente los datos en la tabla y en fechas anteriores.
- **Adicionar:** Permite agregar la demanda al sistema.
- **Modificar:** Permite cambiar los datos previamente ingresados si aún el distribuidor no realizó acciones en ella.
- **Eliminar:** Permite eliminar la demanda si aún el distribuidor no realizó acciones en ella.

Prototipo de Interfaz:

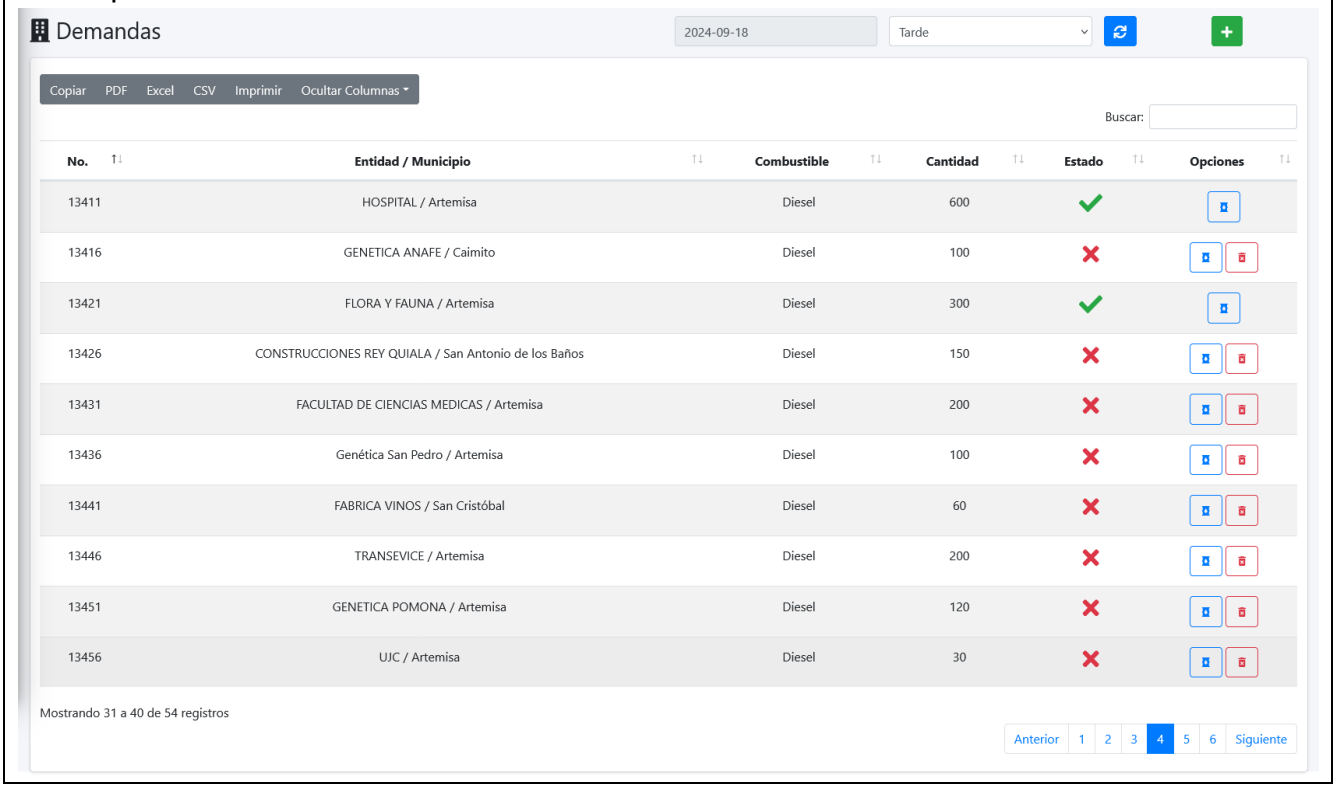


Tabla 13: Historia de Usuario #13. Gestionar Saldo en Tarjeta.

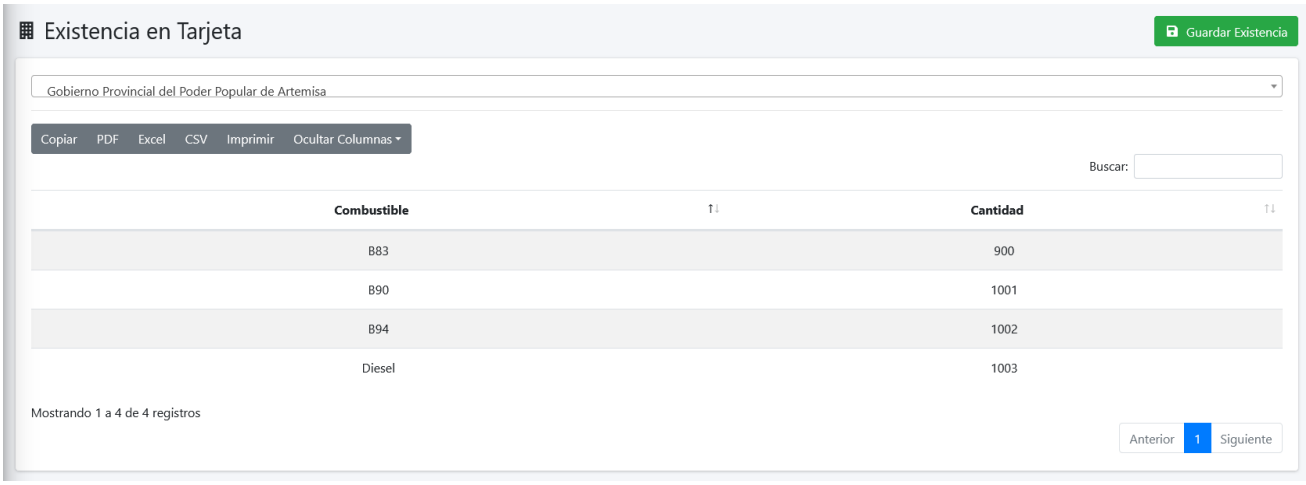
Historias de Usuarios	
Número: HU_13	**Nombre del Requisito**: Gestionar Saldo en Tarjeta
Rol de Usuario: Cliente, Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Media	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión del saldo en tarjeta es donde la entidad podrá conocer lo restante, y el distribuidor será quien actualice el dato. • **Mostrar:** Visualiza las existencias en tarjetas de la entidad. • **Buscar:** Permite buscar dinámicamente los datos en la tabla y en fechas anteriores. • **Adicionar:** Permite agregar la existencia en tarjeta al sistema, con permisos: el administrador y distribuidor. • **Modificar:** Permite cambiar los datos previamente ingresados, con permisos: el administrador y distribuidor. • **Eliminar:** Permite eliminar la existencia en tarjeta, con permisos: el administrador y distribuidor.	
Prototipo de Interfaz: 	

Tabla 14: Historia de Usuario #14. Gestionar Subordinaciones.

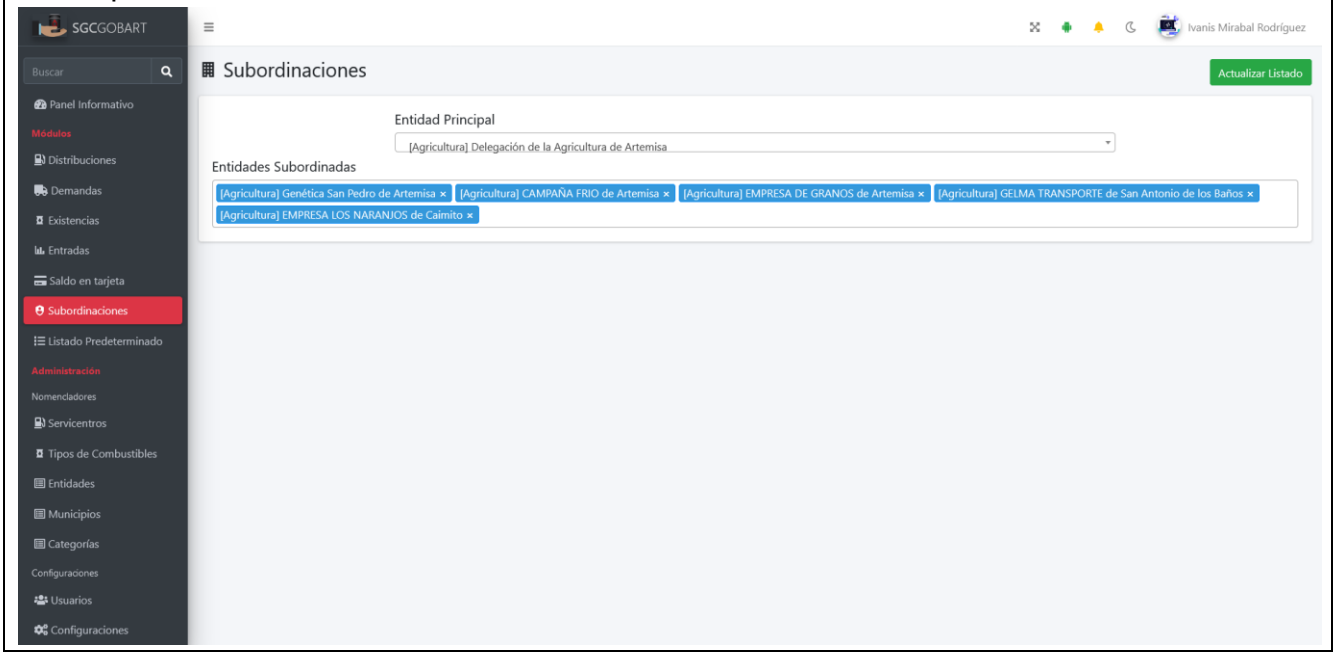
Historias de Usuarios	
Número: HU_14	**Nombre del Requisito**: Gestionar Subordinaciones
Rol de Usuario: Distribuidor, Administrador	**Tiempo estimado**: 0.3
Prioridad: Baja	**Riesgo en desarrollo**: Bajo
Programador: Ivanis Mirabal Rodríguez	
Descripción: La gestión de subordinaciones permite que una entidad pueda ser capaz de visualizar la información de las entidades asignadas a él. • **Seleccionar:** Visualiza un campo donde se seleccionar la entidad a la que se le permitirá la información de las demás. • **Listado:** Al seleccionar la entidad se completará este campo si posee entidades relacionadas, el propio campo es un selector múltiple de opciones, a su vez puede adicionar o eliminar entidades.	
Prototipo de Interfaz: 	

Tabla 15: Historia de Usuario #15. Generar Reportes.

Historias de Usuarios	
Número: HU_15	**Nombre del Requisito**: Generar Reportes
Rol de Usuario: Cliente, Distribuidor, Administrador	**Tiempo estimado**: 1.0
Prioridad: Muy Alta	**Riesgo en desarrollo**: Medio
Programador: Ivanis Mirabal Rodríguez	
Descripción: La generación de reportes es algo crucial para la toma de decisiones. Estos reportes están incluidos visualmente en el Panel Informativo de la web, además cada gestionar según fueron requeridos se les agrego la opción de exportar la información.	
Prototipo de Interfaz:	

Gobierno Provincial del Poder Popular Artemisa

Puesto de Dirección - Distribución de Combustible

Artemisa, 18 de Septiembre de 2024
"Año 66 de la Revolución"

13 de marzo

No	Categoría / Entidad de Municipio	B83	Diesel
1	GEA / GRUPO EMPRESARIAL de Artemisa	0	50

30 de Noviembre

No	Categoría / Entidad de Municipio	Diesel
1	Provincial / AMBULANCIA de Artemisa	150
2	SIN CATEGORIA / UEB MODESTO SERRANO de San Cristóbal	20
3	Provincial / EMPRESA MAYORISTA de San Antonio de los Baños	30
4	Provincial / Lácteo de San Antonio de los Baños	200

Artemisa

No	Categoría / Entidad de Municipio	Diesel
1	Gobierno / CNA TRANSPORTE de Artemisa	205
2	Municipal / COMERCIO TH de Caimito	100
3	Gobierno / Administración Municipal de Guanajay	2700
4	Gobierno / PRESISENTE OLPP de Caimito	50
5	Provincial / Ávicola de Artemisa	1000
6	Provincial / PORCINO de Artemisa	300
7	Provincial / ETECSA de Artemisa	500
8	MININT / GUAJAIBON de Mariel	100
9	Agricultura / CAMPAÑA FRIO de Artemisa	1150
10	Provincial / OBE de Artemisa	200
11	MININT / DELEGACION MININT de Artemisa	100
12	Gobierno / Gobierno Provincial del Poder Popular de Artemisa	350
13	Provincial / EMPRESA MAYORISTA de San Antonio de los Baños	150
14	Provincial / Acueducto de Artemisa	450
15	GEA / GRUPO EMPRESARIAL de Artemisa	1900
16	Provincial / RECURSOS HIDRAULICOS de Artemisa	50
17	Provincial / HOSPITAL de Artemisa	250
18	Provincial / FLORA Y FAUNA de Artemisa	100
19	AZCUBA / TECNOAZUCAR de San Cristóbal	600
20	Escolares / Dirección Provincial Escolares de Artemisa	600
21	Provincial / Gases de Caimito	100
22	Provincial / TRD de Guanajay	100

Desarrollado por: Ivanis Mirabal Rodríguez 18-09-2024

Gobierno Provincial del Poder Popular Artemisa

Puesto de Dirección - Distribución de Combustible

Artemisa, 18 de Septiembre de 2024
"Año 66 de la Revolución"

Delegación de la Agricultura de Artemisa

Desde: 01-09-2024 **Hasta: 18-09-2024**

No.	Servicentro	Municipio	Fecha	Horario	Habilitado
		Diesel			
1	Artemisa	Artemisa	01-09-2024	Mañana	1000
2	Soroa	Candelaria	01-09-2024	Tarde	1000
3	Artemisa	Artemisa	02-09-2024	Mañana	1000
4	Cayo la Rosa	Bauta	02-09-2024	Tarde	1000
5	30 de Noviembre	San Cristóbal	03-09-2024	Mañana	1000
6	Cayo la Rosa	Bauta	03-09-2024	Tarde	500
7	Artemisa	Artemisa	03-09-2024	Tarde	150
8	El Depósito	Güira de Melena	04-09-2024	Mañana	1000
9	30 de Noviembre	San Cristóbal	04-09-2024	Tarde	1000
10	El Depósito	Güira de Melena	05-09-2024	Mañana	1000
11	Artemisa	Artemisa	05-09-2024	Mañana	60
12	Artemisa	Artemisa	06-09-2024	Mañana	60
13	El Depósito	Güira de Melena	07-09-2024	Mañana	1000
14	Artemisa	Artemisa	08-09-2024	Tarde	1000
15	30 de Noviembre	San Cristóbal	10-09-2024	Tarde	800
16	30 de Noviembre	San Cristóbal	11-09-2024	Tarde	600
17	30 de Noviembre	San Cristóbal	13-09-2024	Tarde	500
18	Artemisa	Artemisa	13-09-2024	Tarde	50
19	Artemisa	Artemisa	14-09-2024	Tarde	600
20	Artemisa	Artemisa	16-09-2024	Tarde	100
21	30 de Noviembre	San Cristóbal	16-09-2024	Tarde	600
22	Artemisa	Artemisa	18-09-2024	Tarde	300
		Sub Total:			**14320**

Desarrollado por: Ivanis Mirabal Rodríguez 18-09-2024

Tabla 16: Historia de Usuario #16. Gestionar Notificaciones.

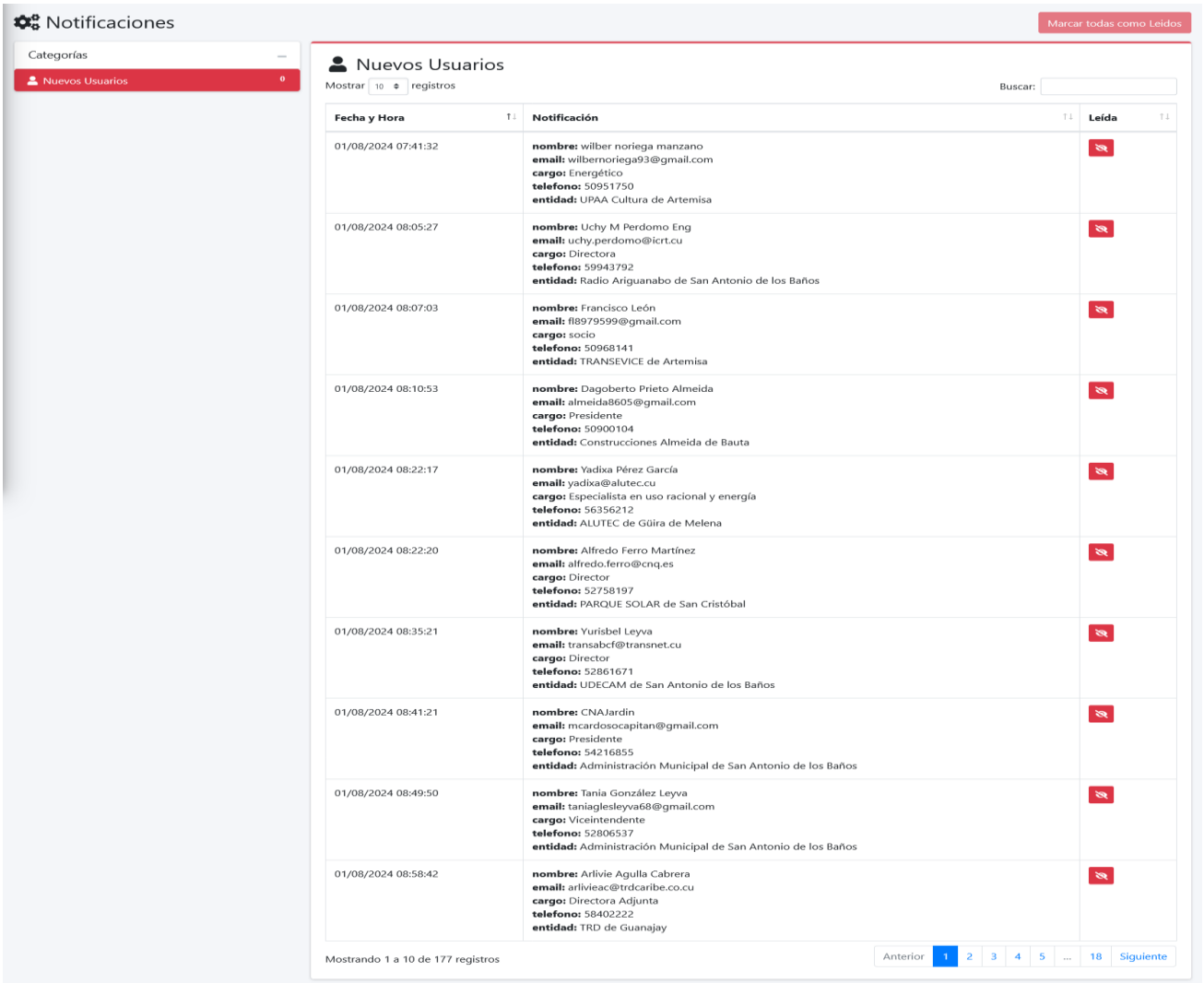
Historias de Usuarios	
Número: HU_16	**Nombre del Requisito**: Gestionar Notificaciones
Rol de Usuario: Cliente, Distribuidor, Administrador	**Tiempo estimado**: 0.4
Prioridad: Baja	**Riesgo en desarrollo**: Bajo
Programador: Ivanis Mirabal Rodríguez	
Descripción: Las notificaciones son un factor relevante para el sistema en cuanto a la ganancia de tiempo, le permite al usuario saber el estado de demandas y asignaciones, a los administradores les permite saber si un nuevo usuario fue registrado y está solicitando acceso.	
Prototipo de Interfaz: 	

Tabla 17: Historia de Usuario #17. Gestionar Listado Predeterminado.

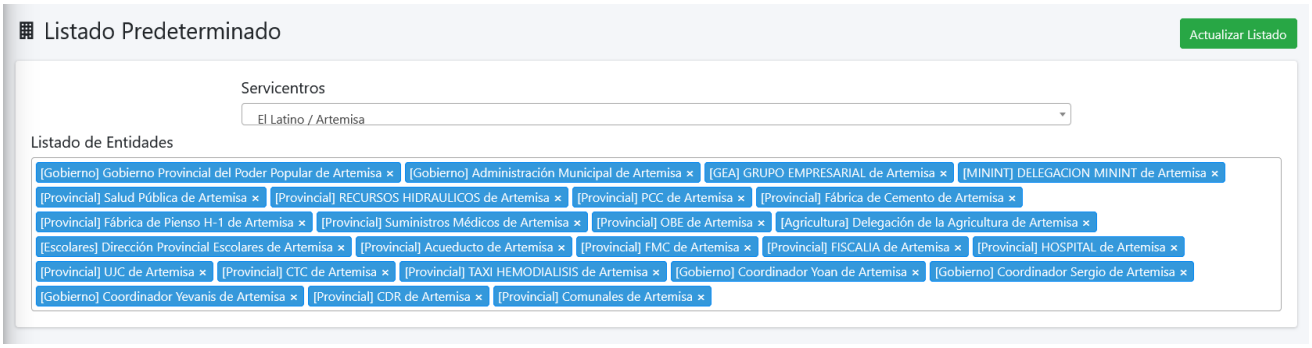
<table>
<tr><th colspan="2">Historias de Usuarios</th></tr>
<tr><td>Número: HU_17</td><td>Nombre del Requisito: Gestionar Listado Predeterminado</td></tr>
<tr><td>Rol de Usuario: Distribuidor, Administrador</td><td>Tiempo estimado: 0.3</td></tr>
<tr><td>Prioridad: Media</td><td>Riesgo en desarrollo: Alta</td></tr>
<tr><td colspan="2">Programador: Ivanis Mirabal Rodríguez</td></tr>
<tr><td colspan="2">Descripción: El gestionar listado predeterminado permite agrupar en un servicentro a las entidades que con mayor frecuencia se les asigna el combustible.
Seleccionar: Visualiza un campo donde se seleccionar el servicentro al que se le agregaran las entidades.
Listado: Al seleccionar el servicentro se completará este campo si posee entidades relacionadas, el propio campo es un selector múltiple de opciones, a su vez puede adicionar o eliminar entidades.</td></tr>
<tr><td colspan="2">Prototipo de Interfaz:
 
</td></tr>
</table>

Tabla 18: Historia de Usuario #18. Visualizar Trazas.

Historias de Usuarios	
Número: HU_18	**Nombre del Requisito**: Visualizar Trazas
Rol de Usuario: Administrador	**Tiempo estimado**: 0.3
Prioridad: Baja	**Riesgo en desarrollo**: Bajo
Programador: Ivanis Mirabal Rodríguez	
Descripción:	
Prototipo de Interfaz:	

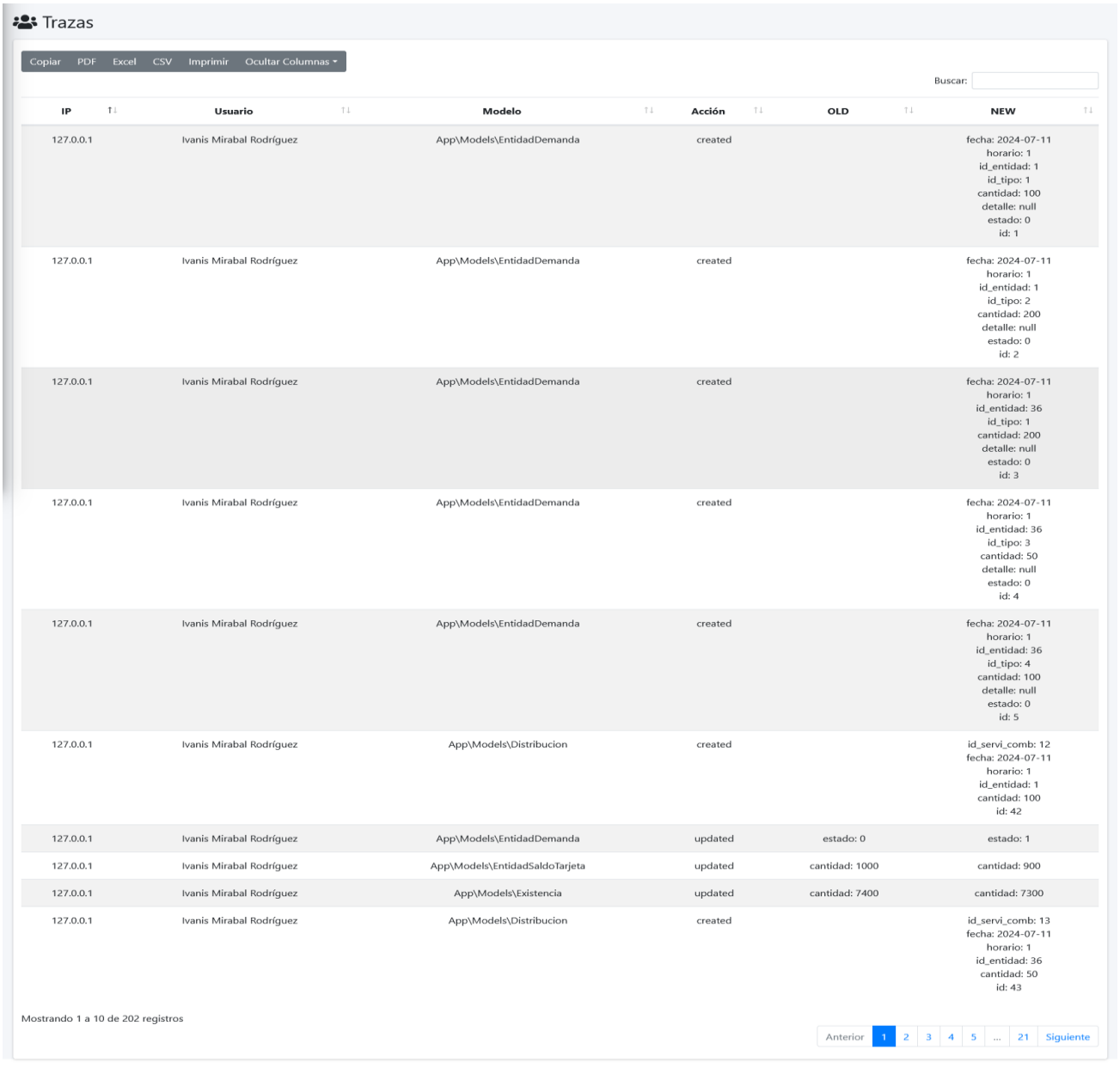


Tabla 19: Historia de Usuario #19. Gestionar Configuración.

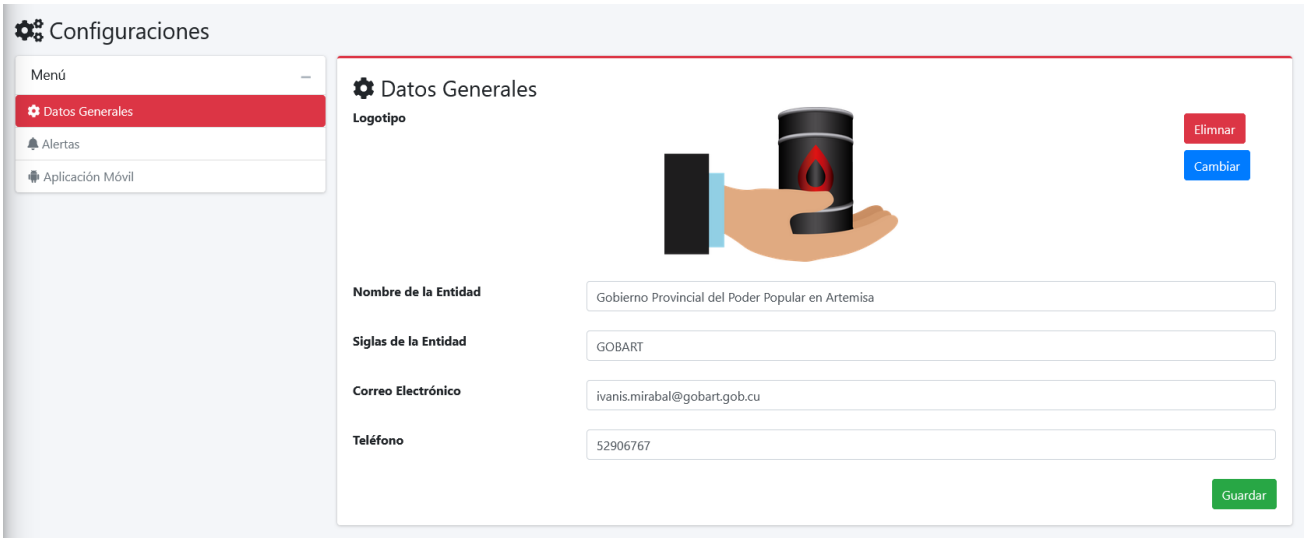
Historias de Usuarios	
Número: HU_19	**Nombre del Requisito**: Gestionar Configuración
Rol de Usuario: Administrador	**Tiempo estimado**: 0.4
Prioridad: Baja	**Riesgo en desarrollo**: Bajo
Programador: Ivanis Mirabal Rodríguez	
Descripción: Permite al Administrador del sistema configurar los datos básicos, como nombre que se mostrara en los reportes, el logotipo y datos de contacto.	
Prototipo de Interfaz: 	

Tabla 20: Historia de Usuario #20. Respaldar Base de Datos.

Historias de Usuarios	
Número: HU_20	**Nombre del Requisito**: Respaldar Base de Datos
Rol de Usuario: Administrador	**Tiempo estimado**: 0.4
Prioridad: Media	**Riesgo en desarrollo**: Bajo
Programador: Ivanis Mirabal Rodríguez	
Descripción: Permite al Administrador del sistema guardar la información de la base de datos de forma independiente o grupal.	
Prototipo de Interfaz:	

Tabla 20. Tarjeta CRC #4. Configuración.

Tarjeta CRC	
Clase: ConfigController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista configuracion. • datos(): Encargado de actualizar los datos del Gobierno como es el logo de la aplicación, número y correo de contactos.	Controllers Request Storage DB Config

Tabla 21. Tarjeta CRC #5. Demanda Entidades.

Tarjeta CRC	
Clase: EntidadDemandaController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista entidad_demanda. • listar(Request $request): Obtiene los datos de la demanda según la información solicitada por el usuario. • buscarServicentro(Request $request): Hace una consulta según la información solicitada por el usuario y devuelve los servicentro con el tipo de combustible solicitado. • cargarDatosServicentro(Request $request): Una vez seleccionado el servicentro se consulta nuevamente para obtener la cantidad de combustible existente. • cargarDatos(Request $request): Obtiene los datos de la demanda y los retorna en formato json para mostrarlos al usuario. • agregar(Request $request): Agrega la distribución en correspon-dencia con la demanda seleccionada. • agregarDemanda(Request $request): Se agrega la demanda. • eliminar(Request $request): Se elimina la demanda en caso de que no hubiese una asignación previa a ella.	Controllers Request EntidadDemanda NrSerComb Distribucion EntidadSaldoTarjeta Existencia NEntidad User NTipoCombustible DB Auth Notification NuevaDemandaNotify Carbon

Tabla 22. Tarjeta CRC #6. Saldo en Tarjeta

Tarjeta CRC	
Clase: EntidadSaldoTarjetaController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista saldo_tarjeta. • listar(Request $request): Obtiene la información según la entidad. • cargarDatos(Request $request): Hace una consulta de la entidad seleccionada y retorna sus datos. • actualizar(Request $request): Actualiza las cantidades existentes en tarjeta.	Controllers Request NEntidad EntidadSaldoTarjeta NTipoCombustible

Tabla 23. Tarjeta CRC #7. Entradas

Tarjeta CRC	
Clase: EntradasController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista entradas. • listar(): Retorna la información de todas las entradas. • eliminar(Request $request): Elimina la entrada de combustible realizando la resta de combustible a la existencia.	Controllers Request Entradas Existencia DB

Tabla 24. Tarjeta CRC #8. Listado Predeterminado

Tarjeta CRC	
Clase: ListadoController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista listado_predeterminado. • getEntidades(Request $request): Obtiene las entidades según el servicentro enviado por parámetro. • actualizar(Request $request): Comprueba las entidades asociadas al servicentro y actualiza la información. • listado(): Muestra la relación de entidades disponibles para ese servicentro. • cargarCombustibles(Request $request): Obtiene los tipos de combustibles disponibles en el servicentro enviado por parámetro. • entidades(): Lista las entidades asociadas al servicentro. • guardar(): Agrega una nueva asignación a las entidades cuya cantidad sea mayor a 0.	Controllers Request NEntidad NrSerComb NServicentro Listado Distribucion EntidadSaldoTarjeta Existencia DB

Tabla 25. Tarjeta CRC #9. Categoría

Tarjeta CRC	
Clase: NCategoriaController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista categoria. • listar(): Obtiene todos los datos de las categorías. • agregar(Request $request): Adiciona una categoría. • cargarDatos(Request $request): Obtiene los datos de la categoría. • actualizar(Request $request): Actualiza la categoría. • eliminar(Request $request): Elimina una categoría.	Controllers Request NCategoria Rule Validator

Tabla 26. Tarjeta CRC #10. Entidad

Tarjeta CRC	
Clase: NEntidadController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista entidad. • listar(): Obtiene todos los datos de las entidades. • agregar(Request $request): Adiciona una entidad. • cargarDatos(Request $request): Obtiene los datos de la entidad. • actualizar(Request $request): Actualiza la entidad. • eliminar(Request $request): Elimina una entidad.	Controllers Request NEntidad Rule Validator NMunicipio NCategoria

Tabla 27. Tarjeta CRC #11. Subordinaciones

Tarjeta CRC	
Clase: NEntidadSubController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista servicentros. • getEntidades(Request $request): Obtiene las entidades relacionadas con la entidad enviada por parámetro. • actualizar(Request $request): Comprueba las entidades relacionadas y actualiza la información.	Controllers Request NEntidad NEntidadSub DB

Tabla 28. Tarjeta CRC #12. Municipio

Tarjeta CRC	
Clase: NMunicipioController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista municipio. • listar(): Obtiene todos los datos de los municipio. • agregar(Request $request): Adiciona un municipio. • cargarDatos(Request $request): Obtiene los datos del municipio • actualizar(Request $request): Actualiza el municipio. • eliminar(Request $request): Elimina un municipio.	Controllers Request NMunicipio Rule Validator

Tabla 29. Tarjeta CRC #13. Notificaciones

Tarjeta CRC	
Clase: NotificacionesController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista notificaciones. • cambiarEstado(Request $request): Cambia el estado de una notificación. • todasLeidas(): Marca todas las notificaciones como leídas. • navbar(): Obtiene las notificaciones no leídas.	Controllers Request Auth

Tabla 30. Tarjeta CRC #14. Tipo de Combustible

Tarjeta CRC	
Clase: NTipoCombustibleController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista tipo_combustible. • listar(): Obtiene todos los datos de los tipos de combustibles. • agregar(Request $request): Adiciona un tipo de combustible. • cargarDatos(Request $request): Obtiene los datos del tipo de combustible. • actualizar(Request $request): Actualiza el tipo de combustible. • eliminar(Request $request): Elimina un tipo de combustible.	Controllers Request NMunicipio Rule Validator

Tabla 31. Tarjeta CRC #15. Reportes

Tarjeta CRC	
Clase: ReportesController	
Responsabilidad	**Colaboración**
• distribución(): Obteniendo por método GET la fecha y el horario de distribución exporta en formato PDF agrupando por servicentros la entidades con las cantidades asignadas. • especifico(Request $request): Exporta en formato PDF las asignaciones realizadas a la entidad seleccionada en el rango de fecha enviado por parámetro. • existencia(): Exporta en formato PDF las existencia por servicentro y tipo de combustible. • entrada(): Exporta en formato PDF todas las entradas según las fechas enviadas por método GET.	Controllers Request PDF Auth Distribucion Existencia NEntidad Entradas

Tabla 32. Tarjeta CRC #16. Respaldar Base de Dato

Tarjeta CRC	
Clase: ReportesController	
Responsabilidad	**Colaboración**
• index(): Retorna la vista respaldo mostrando las tablas de la base de datos. • backupTable(Request $request): Guarda en formato .sql una copia de los datos existentes hasta la fecha.	Controllers Request DB File

Tabla 33. Tarea de ingeniería # 4. Añadir dependencias para roles.

Tarea	
Número de tarea: 4	**Número de HU:** 2
Nombre de la tarea: Añadir dependencias para roles.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 26 de mayo de 2024	**Fecha de fin:** 26 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa Laravel Permission, proporciona la implementación de roles, permisos y relación con los usuarios del sistema.	

Tabla 34. Tarea de ingeniería # 5. Implementar vista de roles.

Tarea	
Número de tarea: 5	**Número de HU:** 2
Nombre de la tarea: Implementar vista de roles.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 26 de mayo de 2024	**Fecha de fin:** 26 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa una vista donde se permita actualizar los roles y a su vez asignarles usuarios.	

Tabla 35. Tarea de ingeniería # 6. Implementar vista de permisos.

Tarea	
Número de tarea: 6	**Número de HU:** 3
Nombre de la tarea: Implementar vista de permisos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 26 de mayo de 2024	**Fecha de fin:** 27 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa una vista donde se permita actualizar los permisos que estarán vinculados a los roles.	

Tabla 36. Tarea de ingeniería # 7. Implementar vista de permisos.

Tarea	
Número de tarea: 7	**Número de HU:** 3
Nombre de la tarea: Implementar vista de permisos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 27 de mayo de 2024	**Fecha de fin:** 27 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se ajusta el sistema a nivel de código para que correspondan los permisos con las rutas accesibles a cada usuario.	

Tabla 37. Tarea de ingeniería # 8. Implementar gestionar categoría.

Tarea	
Número de tarea: 8	**Número de HU:** 4
Nombre de la tarea: Implementar gestionar categoría.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 27 de mayo de 2024	**Fecha de fin:** 28 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de las categorías donde se permitirá la creación, edición y modificación de las mismas.	

Tabla 38. Tarea de ingeniería # 9. Implementar gestionar municipio.

Tarea	
Número de tarea: 9	**Número de HU:** 5
Nombre de la tarea: Implementar gestionar municipio.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 28 de mayo de 2024	**Fecha de fin:** 29 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de las categorías donde se permitirá la creación, edición y modificación de las mismas.	

Tabla 39. Tarea de ingeniería # 10. Implementar gestionar entidad.

Tarea	
Número de tarea: 10	**Número de HU:** 6
Nombre de la tarea: Implementar gestionar entidad.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 29 de mayo de 2024	**Fecha de fin:** 30 de mayo de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de las entidades donde se permitirá la creación, edición y modificación de las mismas.	

Tabla 40. Tarea de ingeniería # 12. Implementar gestionar servicentro.

Tarea	
Número de tarea: 12	**Número de HU:** 8
Nombre de la tarea: Implementar gestionar servicentro.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 31 de mayo de 2024	**Fecha de fin:** 1 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de los servicentros donde se permitirá la creación, edición y modificación de las mismas.	

Tabla 41. Tarea de ingeniería # 13. Implementar vista de existencia.

Tarea	
Número de tarea: 13	**Número de HU:** 9
Nombre de la tarea: Implementar vista de existencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 1 de junio de 2024	**Fecha de fin:** 2 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de las existencias, listando todos los servicentros separándolos por tipos de combustible.	

Tabla 42. Tarea de ingeniería # 14. Implementar la funcionalidad de existencia.

Tarea	
Número de tarea: 14	**Número de HU:** 9
Nombre de la tarea: Implementar la funcionalidad de existencia.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 2 de junio de 2024	**Fecha de fin:** 3 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se permitirá la adición de cantidades de combustible lo que sumará a la existencia y actualizará el número de la reserva.	

Tabla 43. Tarea de ingeniería # 15. Implementar gestionar entradas.

Tarea	
Número de tarea: 15	**Número de HU:** 10
Nombre de la tarea: Implementar gestionar entradas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 3 de junio de 2024	**Fecha de fin:** 4 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de las entradas donde se permitirá solo la eliminación de las mismas.	

Tabla 44. Tarea de ingeniería # 16. Implementar vista de distribución.

Tarea	
Número de tarea: 16	**Número de HU:** 11
Nombre de la tarea: Implementar gestionar entradas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 4 de junio de 2024	**Fecha de fin:** 5 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de las distribuciones donde se permitirá realizar todas las acciones sobre las distribuciones.	

Tabla 45. Tarea de ingeniería # 17. Implementar funcionalidad de distribución.

Tarea	
Número de tarea: 17	**Número de HU:** 11
Nombre de la tarea: Implementar funcionalidad de distribución.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 5 de junio de 2024	**Fecha de fin:** 7 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la lógica de la distribución donde al distribuir a un organismo se debe restar de su existencia en tarjeta y por consiguiente debe ser restado de los niveles de existencia en ese servicentro.	

Tabla 46. Tarea de ingeniería # 18. Implementar distribución por listado predeterminado.

Tarea	
Número de tarea: 18	**Número de HU:** 11
Nombre de la tarea: Implementar distribución por listado predeterminado.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 7 de junio de 2024	**Fecha de fin:** 9 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de la distribución por listado predeterminado donde permitirá al distribuidor realizar en una misma vista más de una distribución disminuyendo el tiempo de trabajo.	

Tabla 47. Tarea de ingeniería # 19. Implementar vista de demanda.

Tarea	
Número de tarea: 19	**Número de HU:** 12
Nombre de la tarea: Implementar vista de demanda.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 11 de junio de 2024	**Fecha de fin:** 12 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista de las demandas permitiendo solamente su realización en el horario establecido por el cliente, permite realizar la demanda en todos los tipos de combustible 1 sola vez durante el horario establecido.	

Tabla 48. Tarea de ingeniería # 20. Implementar subordinaciones a las demandas.

Tarea	
Número de tarea: 20	**Número de HU:** 12
Nombre de la tarea: Implementar subordinaciones a las demandas.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 12 de junio de 2024	**Fecha de fin:** 14 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa que el select de las entidades verifique si la entidad del usuario tiene relación con otras entidades de ser así se le facilita que también pueda realizar la demanda por dicha entidad.	

Tabla 49. Tarea de ingeniería # 21. Implementar gestionar saldo en tarjeta.

Tarea	
Número de tarea: 21	**Número de HU:** 13
Nombre de la tarea: Implementar gestionar saldo en tarjeta.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.1
Fecha de inicio: 14 de junio de 2024	**Fecha de fin:** 15 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementa la funcionalidad de actualizar el saldo existente en la tarjeta de las entidades del sistema por cada uno de los tipos de combustible.	

Tabla 50. Tarea de ingeniería # 22. Implementar gestionar subordinaciones.

Tarea	
Número de tarea: 22	**Número de HU:** 14
Nombre de la tarea: Implementar gestionar subordinaciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 15 de junio de 2024	**Fecha de fin:** 17 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla la vista donde permitirá vincular varias entidades a una entidad principal.	

Tabla 51. Tarea de ingeniería # 25. Implementar gestionar notificaciones.

Tarea	
Número de tarea: 25	**Número de HU:** 16
Nombre de la tarea: Implementar gestionar notificaciones.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha de inicio: 20 de junio de 2024	**Fecha de fin:** 22 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementará una notificación en la barra superior del sistema que muestre el número de notificaciones sin leer del usuario, además se enviaran correos electrónicos, incluye el desarrollo de una vista que permita al usuario ver los detalles y marcar como leída.	

Tabla 52. Tarea de ingeniería # 26. Implementar gestionar listado predeterminado.

Tarea	
Número de tarea: 26	**Número de HU:** 17
Nombre de la tarea: Implementar gestionar listado predeterminado.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 22 de junio de 2024	**Fecha de fin:** 25 de junio de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementará una vista para crear la relación de un servicentro con entidades, permite la distribución de forma masiva en la sección de distribución.	

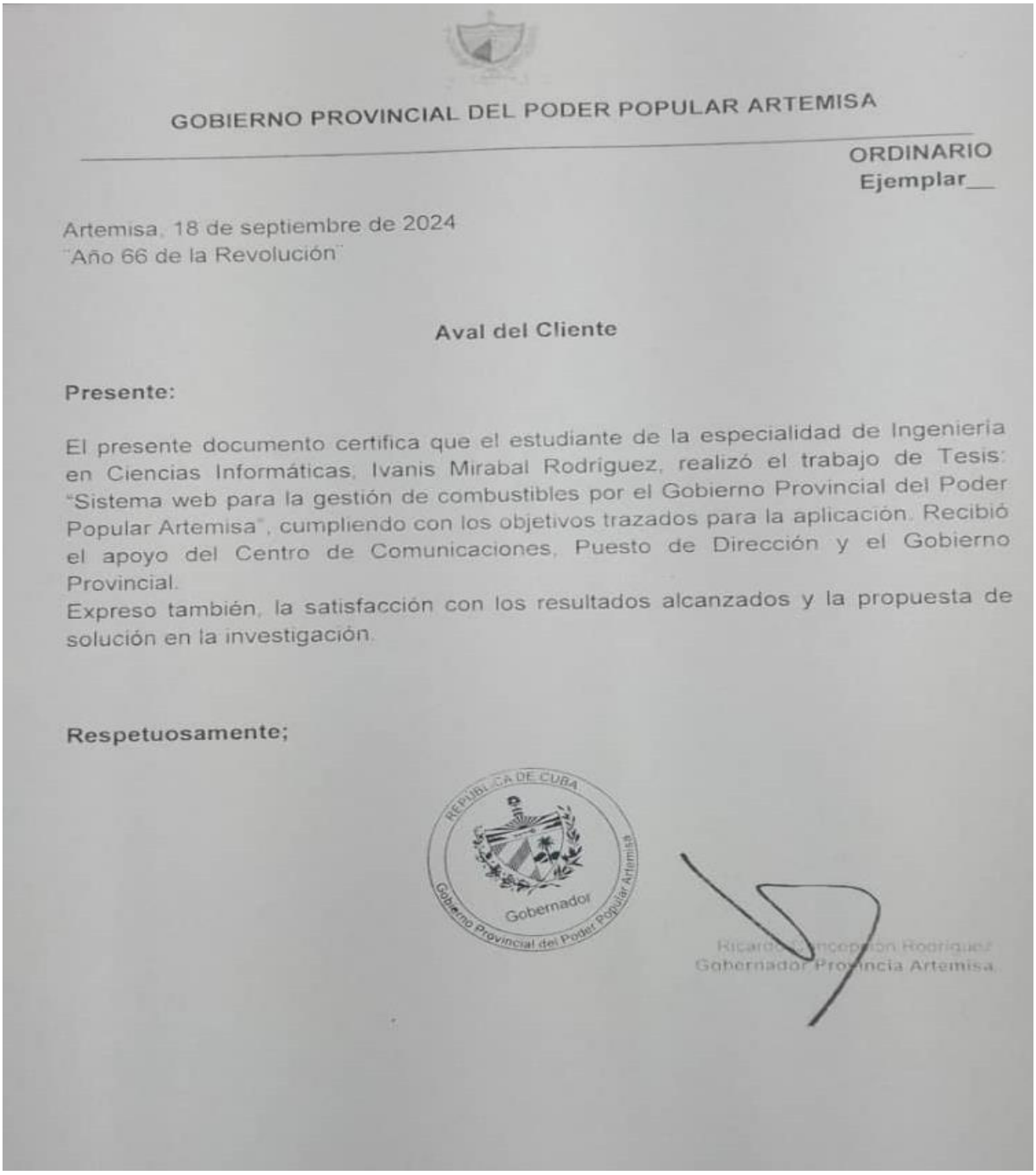
Tabla 53. Tarea de ingeniería # 28. Implementar gestionar configuración.

Tarea	
Número de tarea: 28	**Número de HU:** 19
Nombre de la tarea: Implementar gestionar configuración.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 2 de agosto de 2024	**Fecha de fin:** 5 de agosto de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se implementará la vista para la actualización de los datos del sistema y datos de contacto, se ajustan los reportes y demás secciones de la web para que obtengan el logotipo especificado en la configuración.	

Tabla 54. Tarea de ingeniería # 29. Implementar respaldo de base de datos.

Tarea	
Número de tarea: 29	**Número de HU:** 20
Nombre de la tarea: Implementar respaldo de base de datos.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.3
Fecha de inicio: 2 de agosto de 2024	**Fecha de fin:** 5 de agosto de 2024
Programador responsable: Ivanis Mirabal Rodríguez	
Descripción: Se desarrolla una vista que muestra las tablas del sistema y la cantidad de registros por cada uno, se permite exportar los datos a una carpeta interna del sistema para su posterior recuperación en caso de que el sistema sufra perdida de datos.	

AVAL DEL CLIENTE



GOBIERNO PROVINCIAL DEL PODER POPULAR ARTEMISA

ORDINARIO
Ejemplar__

Artemisa, 18 de septiembre de 2024
"Año 66 de la Revolución"

Aval del Cliente

Presente:

El presente documento certifica que el estudiante de la especialidad de Ingeniería en Ciencias Informáticas, Ivanis Mirabal Rodríguez, realizó el trabajo de Tesis: "Sistema web para la gestión de combustibles por el Gobierno Provincial del Poder Popular Artemisa", cumpliendo con los objetivos trazados para la aplicación. Recibió el apoyo del Centro de Comunicaciones, Puesto de Dirección y el Gobierno Provincial.
Expreso también, la satisfacción con los resultados alcanzados y la propuesta de solución en la investigación.

Respetuosamente;

REPÚBLICA DE CUBA
Gobernador
Gobierno Provincial del Poder Popular Artemisa

Ricardo [illegible] Concepción Rodríguez
Gobernador Provincia Artemisa