



“2022. Año del Quincentenario de la Fundación de Toluca de Lerdo, Capital del Estado de México”.

TECNOLÓGICO DE ESTUDIOS SUPERIORES DEL ORIENTE DEL ESTADO DE MÉXICO

“MATZY SYSTEM”

MEMORIA DE RESIDENCIA PROFESIONAL

PARA OBTENER EL TÍTULO DE:

INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTA:

JUAREZ ANDRADE ANTONIO

LOS REYES ACAQUILPAN, ESTADO DE MÉXICO 2022

ASESOR METODOLÓGICO: VALENCIA CRUZ OMAR

ASESOR TÉCNICO: NAVA HERNÁNDEZ GERARDO

HOJA DE APROBACIÓN DE MEMORIA RESIDENCIA PROFESIONAL.

DEDICATORIAS Y/O AGRADECIMIENTOS.

ÍNDICE.

Contenido

INTRODUCCIÓN.....	6
JUSTIFICACIÓN	7
OBJETIVOS	7
CAPITULO I: CULTURA ORGANIZACIONAL DE LA CONSTRUCTORA MATZY	8
1.1 ANTECEDENTES.....	9
1.2 FILOSOFÍA DE LA EMPRESA.....	9
1.2.1 MISIÓN.....	9
1.2.2 VISIÓN	9
1.2.3 VALORES.....	9
1.2.4 OBJETIVOS DE LA EMPRESA	10
1.3 ORGANIGRAMA GENERAL.....	10
1.3.1 PARTES DEL ORGANIGRAMA.....	10
CAPITULO II: ANÁLISIS DEL SISTEMA ACTUAL.....	12
2.1 ANÁLISIS DEL SISTEMA ACTUAL	13
2.1.1 SISTEMA ACTUAL.....	13
2.1.2 BREVE DESCRIPCIÓN GENERAL DEL SISTEMA.....	13
2.1.3 SERVICIOS QUE PROPORCIONA LA EMPRESA.....	13
2.1.4 EQUIPAMIENTO CON EL QUE CUENTA.....	14
2.1.5 CLIENTES.....	14
2.1.6 MANEJO DE PAGOS Y SERVICIOS.....	14
2.2 DIAGRAMA DE PROCESO DEPARTAMENTO DE SERVICIOS.....	15
2.2.1 FIGURAS DE PROCESO REPRESENTANTE LEGAL	16
2.2.2 FIGURAS DE PROCESO PARA SERVICIOS.....	17
2.2.3 FIGURAS DE PROCESO PARA CLIENTE /EMPRESA.....	17
2.2.4 REPORTE HECHO POR LA REPRESENTANTE LEGAL PARA EL DEPARTAMENTO DE SERVICIOS.....	18
2.3 DIAGRAMA DE PROCESO PARA ASIGNACIÓN DE PRESUPUESTO A UN SERVICIO.....	19
2.3.1 FIGURAS DE PROCESO PARA EL REPRESENTANTE LEGAL.....	20
2.3.2 FIGURAS DE PROCESO PARA INICIO DE SERVICIO.....	20

2.3.3 FIGURAS DE PROCESO PARA LAS FINANZAS.	21
2.3.4 REPORTE HECHO POR LA REPRESENTANTE LEGAL PARA EL DEPARTAMENTO DE FINANZAS.	21
CAPITULO III: ELECCIÓN DE LA METODOLOGÍA.	22
3.1 ELECCIÓN DE LA METODOLOGÍA.	23
3.1.1 METODOLOGÍAS TRADICIONALES.	23
3.1.2 METODOLOGÍAS AGILES	26
3.2 ELECCIÓN DE LA METODOLOGÍA	31
CAPITULO IV: APLICACIÓN DE LA METODOLOGÍA SCRUM MODIFICADO (MÉTODO AXOLOTL)	32
4.1 APLICACIÓN DE LA METODOLOGÍA	33
4.1.1 BREIF	33
4.2 REQUISITOS	34
4.3 PRODUCT BACKLOG	34
4.4 TABLERO DE RESPONSABILIDADES	36
4.5 DESARROLLO	36
SPRINT BACKLOG SP1 H1	36
SPRINT BACKLOG SP2 H2	55
SPRINT BACKLOG SP3 H3	69
SPRINT BACKLOG SP4 H4	105
4.6 RESULTADOS OBTENIDOS	131
DOCUMENTO DE ACEPTACIÓN DE LA ENTREGA TOTAL DEL SOFTWARE	134
CARTA DE AGRADECIMIENTO	136
4.7 CONCLUSIONES	137
FUENTES DE INFORMACIÓN	138
5.1 REFERENCIAS DIGITALES	139
GLOSARIO	141
6.1 GLOSARIO TÉCNICO	142

INTRODUCCIÓN

Capítulo I cultura organizacional de la constructora Matzy

La Empresa Constructora Matzy S.A. de C.V. es una empresa dedicada al alquiler de maquinaria de construcción y vehículos de carga, dicha empresa actualmente no cuenta con ningún sistema o herramienta que permita automatizar las operaciones administrativas ya que todos sus trámites son manuales y el seguimiento del servicio no es el adecuado, es por ello que se dio a la tarea de desarrollar un software administrativo para que la empresa llevé un registro de sus clientes, equipos y finanzas.

Capítulo II análisis del sistema actual

La empresa no cuenta con ningún sistema de apoyo para el flujo de trabajo que le facilite brindar un servicio y saber lo que tiene disponible al momento, lo único considera como registro son notas mentales y escritas en cuadernos a manera de recordatorio para saber el estado actual de sus vehículos, motivo por el cual se le sugirió un sistema donde pueda llevar un registro de todo lo que sucede en dicha empresa para tener una optimización de su información y horas de trabajo.

Capítulo III elección de la metodología

La elección de la metodología de trabajo y de desarrollo debe de presentar muchas ventajas y flujos de trabajo rápidos, se hace un análisis para determinar que metodología es la más apropiada para el proyecto en cuestión, pasando por metodologías tradicionales y modernas.

Capítulo IV aplicación de la metodología

A través de los conocimientos adquiridos a lo largo de la carrera de ingeniería en sistemas computacionales y las herramientas conocidas por el sistema DUAL, se decidió hacer uso de las tecnologías de desarrollo web tales como PHP, CSS, HTML y JavaScript, mediante el framework Laravel, siguiendo el patrón de arquitectura de software MVC, además de tecnologías de diseño como Adobe XD y Adobe Photoshop bajo estructuras de diseño UX y UI, apoyando el desarrollo del proyecto en una variante de la metodología ágil conocida como Scrum.

JUSTIFICACIÓN

La empresa no cuenta con un sistema o métodos para optimizar sus funciones del día a día, por lo que gran parte de sus registros se pierden entre notas en papel y quizás nunca sean registrados o consultados esto provoca el duplicado de información.

Dando así origen a un sistema que automatiza la mayoría de las acciones involucradas en la realización de un nuevo registro, el cual incluye datos como cliente, gastos, ingresos, fechas, recursos, disponibilidades y así beneficiar a la empresa de esta manera durante la permanencia en la misma mientras se aprende nuevas habilidades y conocimientos.

OBJETIVOS

Objetivo general

Desarrollar un software web de tipo administrativo que permita el acceso desde cualquier lugar y dispositivo con acceso a internet, generar reportes internos dentro de la empresa para los diferentes departamentos. El software tiene que ser fácil de usar y que cumpla las expectativas esperadas de la empresa optimizando su flujo de trabajo.

Objetivos específicos

- Analizar el flujo de trabajo actual, cuyo principio es registrar únicamente a los clientes y los vehículos que ocupan, los cuales se registran en notas manuscritas.
- Implementar un sistema con base en la programación, y así llevar el flujo de trabajo a procesos óptimos y automatizados que permitan facilitar las tareas de registros de datos.
- Realizar pruebas unitarias del sistema con datos reales de la empresa, con la finalidad de hacer el sistema lo más preciso posible y así cumplir las expectativas del cliente.
- Desarrollar un sistema que se pueda visualizar desde cualquier dispositivo para garantizar la portabilidad del mismo.

CAPITULO I: CULTURA ORGANIZACIONAL DE LA CONSTRUCTORA MATZY

1.1 ANTECEDENTES.

La constructora Matzy es una empresa de capital variable dedicada al alquiler de maquinaria de construcción y vehículos para su transporte, tiene su domicilio en Calle Mil Cumbres #1, Colonia Santo Tomás, Municipio de Ixtapaluca, Estado de México y opera en toda la República Mexicana de conformidad con las necesidades de su cliente, llevando sus máquinas a donde las necesite, cumpliendo trabajos como:

- Limpieza de terrenos
- Aplanado de terrenos
- Demolición de estructuras
- Transporte de vehículos
- Alquiler de máquinas de construcción
- Alquiler de vehículos de transporte

Siendo estos los servicios que pone a disposición de sus clientes, siendo su principal giro de negocio el alquiler.

1.2 FILOSOFÍA DE LA EMPRESA.

1.2.1 MISIÓN

Proveer equipos para medianas y grandes empresas que requieren equipo de maquinaria pesada cuando no cuentan con el equipo necesario para solucionar sus problemas y mantener un trato confiable durante todo el servicio con los clientes.

1.2.2 VISIÓN

Constructora Matzy aspira a ser un referente en cuanto a servicios de construcción se refiere, ayudando al mayor número posible de clientes a culminar su obra y todo ello con un servicio y un equipo humano de calidad, útil y eficiente que aporte una solución certera a los problemas planteadas por los clientes.

1.2.3 VALORES

Los valores de la empresa serán siempre la puntualidad en sus servicios, la calidad de los equipos, la confianza con los clientes, reconociéndolos a todos como socios a la hora de ayudar a que sus proyectos se materialicen tal y como lo han planificado.

1.2.4 OBJETIVOS DE LA EMPRESA

Objetivo general

El objetivo general es mantener siempre una clientela satisfecha que el día de mañana sean ellos quien regresen a solicitar un nuevo servicio por lo impecable que fue el servicio brindado.

Objetivos específicos

- Mantener la confianza del cliente
- Ofrecer servicios a nivel nacional
- Proporcionar un servicio y equipo de calidad

1.3 ORGANIGRAMA GENERAL

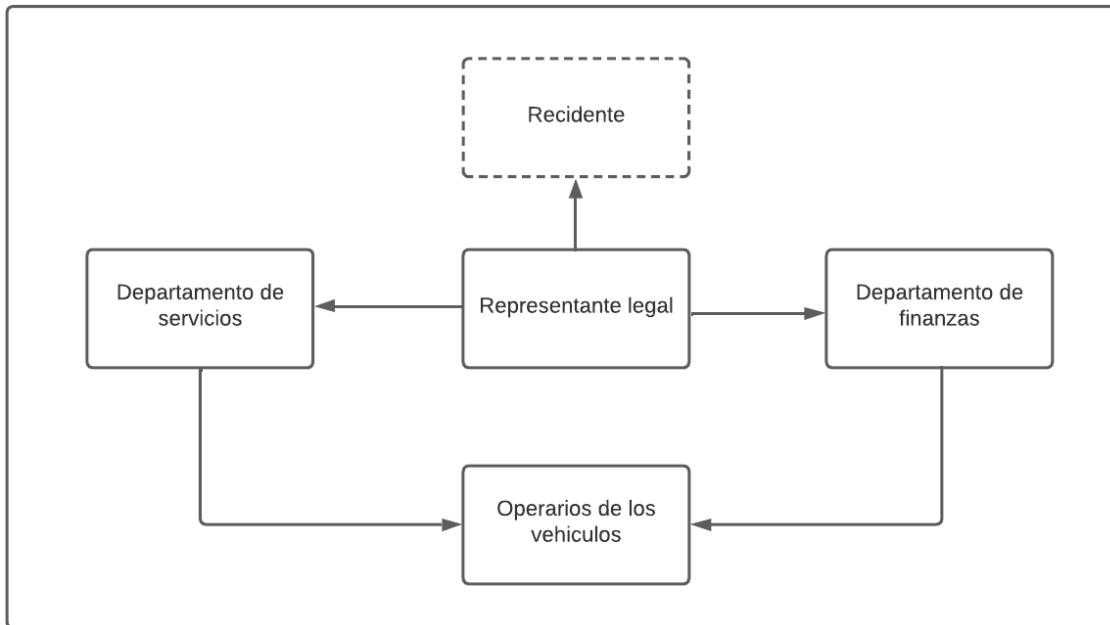


Figura 1.1 Organigrama de la empresa Constructora Matzy

1.3.1 PARTES DEL ORGANIGRAMA Representante legal

El representante legal es el dueño de la empresa, papel que funge la Lic. Mónica Guerrero Pérez.

Departamento de servicios

Este departamento se dedica a la comunicación directa con los clientes y se encarga de prestar o no un servicio según sus condiciones.

Departamento de finanzas

Este departamento se dedica a llevar la contabilidad de la empresa, los gastos, ingresos y ganancias que obtiene la empresa.

Operarios de los vehículos

Los operadores de vehículos son personas capacitadas para operar tanto vehículos de transporte como máquinas de construcción.

Residente

Es su servidor que realiza su sistema dual.

CAPITULO II: ANÁLISIS DEL SISTEMA ACTUAL.

2.1 ANÁLISIS DEL SISTEMA ACTUAL

2.1.1 SISTEMA ACTUAL.

Actualmente la empresa Constructora Matzy S.A. de C.V. No cuenta con ningún sistema de registro de sus vehículos o herramientas que facilite el conteo de horas, materiales o servicios que brinda la empresa, ya que, en palabras del representante legal, Lic. Mónica Guerrero Pérez, “lo único que se considera como registro son notas mentales y escritas en cuadernos a modo de recordatorios” para conocer el estado actual de sus vehículos, por lo que necesita una herramienta que le permita realizar todos estos registros y mantenerlos organizados. Existen dos vertientes en cuanto a sus vehículos las llamadas Maquinas, son en concreto 4 del tipo de máquinas para excavación y aplanado de terreno, también cuenta con dos tractores y sus correspondientes estructuras de carga de tipo double drop deck, estos vehículos son más conocidos como tráiler de carga y llamados por la empresa como palas, la manera en la que la empresa tiene estos vehículos registrados para poder mencionarlos es a través de su nombre y placa o matricula.

2.1.2 BREVE DESCRIPCIÓN GENERAL DEL SISTEMA.

Debido a las necesidades de la empresa y el tipo de servicios que brinda a nivel nacional, se tomó la decisión de crear un software o sistema web para que pueda gestionar todo desde cualquier lugar o dispositivo, esto se llevará a cabo mediante el uso de un framework llamado Laravel, que se adapta perfectamente a las necesidades de la empresa, actualmente cuenta con dos módulos principales que cubren las necesidades de “servicios” y “finanzas”. Cabe señalar que el software se desarrolla paso a paso bajo lo que el cliente requería y siendo revisado las veces que sea necesario por el mismo, todo bajo lo que propone el método “Lean”, que dice que: “La producción debe estar basada en la demanda y no ofrecer”.

2.1.3 SERVICIOS QUE PROPORCIONA LA EMPRESA.

Los servicios prestados por Constructora Matzy S.A. de C.V. son para el alquiler o transporte de máquinas y vehículos de construcción para empresas u organismos externos que requieran de estas herramientas, sin embargo, la empresa actualmente no

lleva ningún registro ni promoción de los servicios prestados ya que al ser una empresa relativamente nueva carece de procesos o sistema para ello, existen dos servicios de alquiler y dependen del tipo de vehículo alquilado.

2.1.4 EQUIPAMIENTO CON EL QUE CUENTA.

Existen dos tipos de vehículos, las máquinas de construcción, son específicamente cuatro máquinas para excavación y allanamiento de terrenos, también cuenta con dos tractores y sus correspondientes estructuras de carga tipo double drop deck, estos vehículos son más conocidos como remolques de carga y llamados por la empresa como palas.

2.1.5 CLIENTES.

Los clientes o empresas solicitan un servicio de alquiler a la constructora, el representante legal verifica la disponibilidad de los vehículos requeridos para su proyecto.

2.1.6 MANEJO DE PAGOS Y SERVICIOS.

Una vez que la empresa aprueba el servicio al cliente, el cliente entrega un enganche que es proporcional al costo de la renta de los vehículos asignados y este ingreso es registrado por el representante legal bajo el concepto de enganche, esto se denomina asignación de presupuesto a un servicio, ingresos y gastos se imputan a este presupuesto según sea el caso y a través de estos se obtienen las ganancias del servicio de alquiler.

2.2 DIAGRAMA DE PROCESO DEPARTAMENTO DE SERVICIOS.

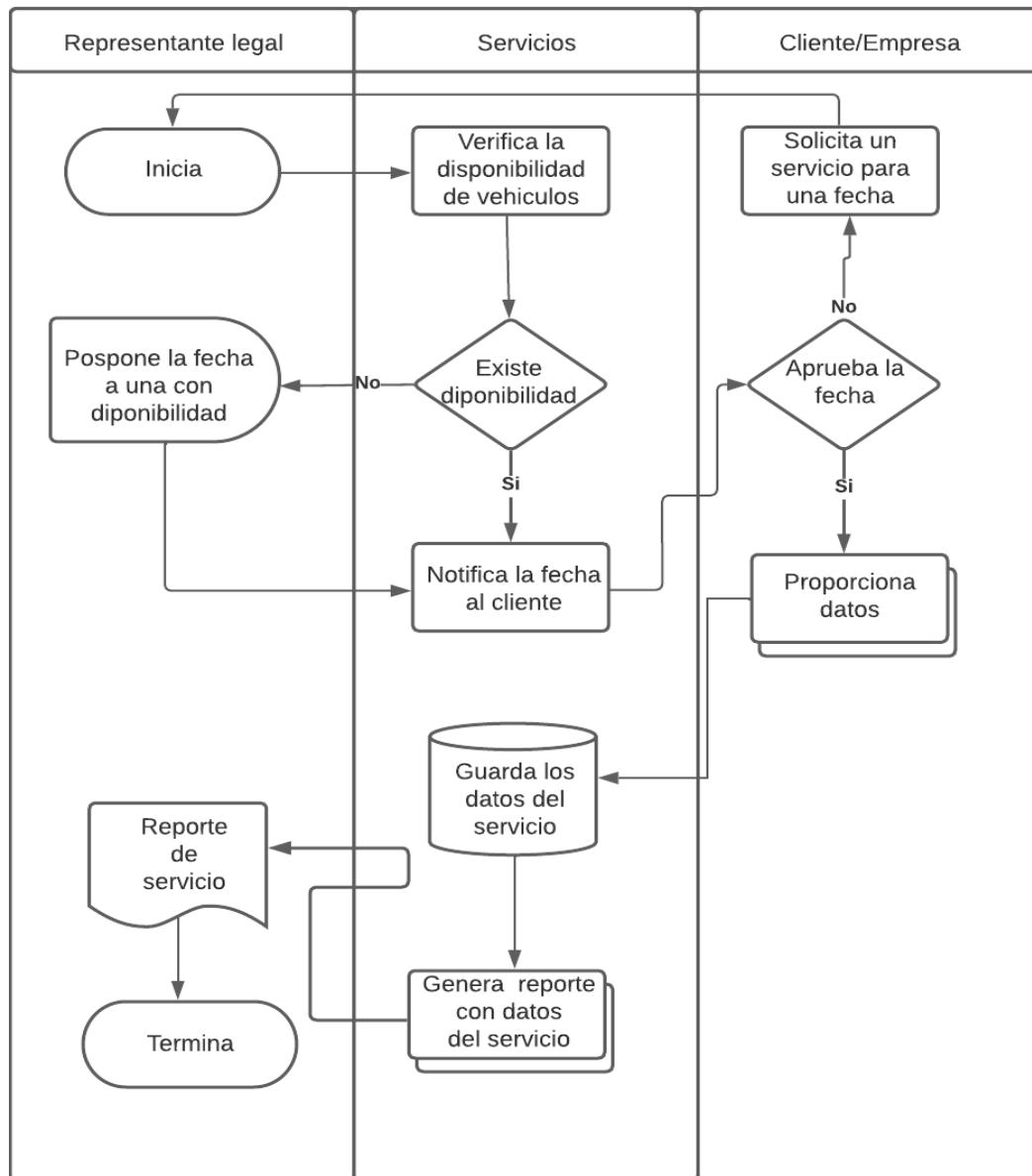


Figura 2.1 Diagrama de proceso de servicios

2.2.1 FIGURAS DE PROCESO REPRESENTANTE LEGAL.

Figura	Significado de la figura	Entidad	Ejecución
Inicia	Inicio o termino de un proceso	Representante legal	Punto en el que el representante legal inicia un servicio para un empresa o cliente
Pospone la fecha a una con disponibilidad	Retardo	Representante legal	La fecha se pospone debido a que no existe disponibilidad de esta ocasionando un retardo en el proceso
Reporte de servicio	Documento	Representante legal	Donde se genera la caída de información a un formato físico o digital que aprueba un servicio iniciado
Termina	Inicio o termino de un proceso	Representante legal	Punto en el que termina el proceso para iniciar el servicio y dar pie al siguiente paso

Figura 2.2 Descripción de las figuras representante legal.

2.2.2 FIGURAS DE PROCESO PARA SERVICIOS.

Figura	Significado de la figura	Entidad	Ejecución
	Proceso	Servicios	Se verifica la disponibilidad de los vehículos y en que momento estos estarán disponibles
	Decisión	Servicios	Se toma una decisión para saber si la fecha que solicita el cliente está disponible o si de lo contrario se pospone
	Proceso	Servicios	Se notifica al cliente de a través de una llamada telefónica, mensaje o correo según el medio de comunicación de preferencia
	Base de datos	Servicios	Punto en el que se almacenan los datos del cliente y vehículos que este requiera
	Proceso dependiente de otro proceso	Servicios	Proceso dependiente de la disponibilidad de la empresa y la aceptación del cliente para la generación de un reporte

Figura 2.2 Descripción de las figuras de servicios.

2.2.3 FIGURAS DE PROCESO PARA CLIENTE /EMPRESA.

Figura	Significado de la figura	Entidad	Ejecución
	Proceso	Cliente/Empresa	Punto de partida para el inicio de proceso a través de la solicitud del cliente
	Decisión	Cliente/Empresa	El cliente aprueba la fecha ya que esta puede ser la que el solicitó o alguna que la empresa propone
	Proceso dependiente de otro proceso	Cliente/Empresa	Proceso dependiente de la disponibilidad y la solicitud del cliente para que el mismo proporcione sus datos e iniciar el servicio

Figura 2.3 Descripción de las figuras cliente/empresa.

2.2.4 REPORTE HECHO POR LA REPRESENTANTE LEGAL PARA EL DEPARTAMENTO DE SERVICIOS.

Nombret	Yolanda	Dia	Mes	Año	Folio
Tema:	Datos del servicio	15	08	2020	167

Datos del cliente/empresa
Bodegas temporales S.A DE C.V.

Dirección: Zacatecas, Colonia [REDACTED]

Teléfono: [REDACTED]

Vehículos:
Excavadora de arreglos 15-08-2020 al 01-11-2020

Pala del 15 al 22 -2020

Mantenimiento de la máquina cada 400 hrs.

Figura 2.4 Registro de un nuevo servicio, forma de trabajo actual.

2.3 DIAGRAMA DE PROCESO PARA ASIGNACIÓN DE PRESUPUESTO A UN SERVICIO.

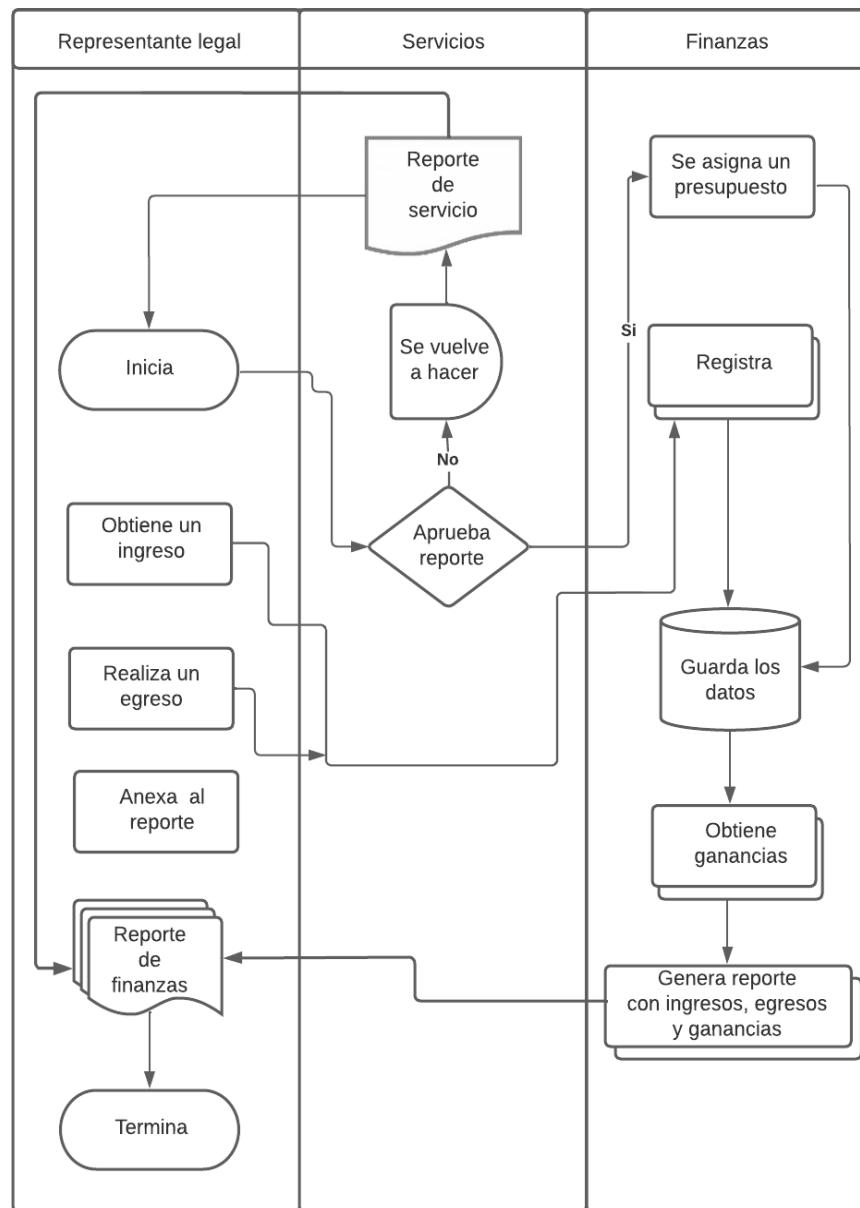


Figura 2.5 Diagrama de proceso para finanzas.

2.3.1 FIGURAS DE PROCESO PARA EL REPRESENTANTE LEGAL.

Figura	Significado de la figura	Entidad	Ejecución
Inicia	Inicio o termino de un proceso	Representante legal	Punto en el que el representante legal inicia un presupuesto para un servicio.
Obtiene un ingreso	Retardo	Representante legal	Obtiene un ingreso de un servicio, es un pago por parte del cliente
Realiza un egreso	Retardo	Representante legal	Realiza un egreso de un servicio, esto indica un gasto por parte de la empresa
Anexa al reporte	Documento	Representante legal	Se anexa el reporte de servicios para que el reporte de presupuesto se genere con el
Reporte de finanzas	Documentos	Representante legal	Donde se genera la caída de información, formato físico o digital donde se muestra tanto el reporte de servicios como el de presupuesto
Termina	Inicio o termino de un proceso	Representante legal	Se termina el proceso.

Figura 2.6 Descripción de las figuras del representante legal.

2.3.2 FIGURAS DE PROCESO PARA INICIO DE SERVICIO.

Figura	Significado de la figura	Entidad	Ejecución
Aprueba reporte	Decisión	Servicios	Aprueba el reporte, esto implica que existe primero un servicio.
Se vuelve a hacer	Retardo	Servicios	Se vuelve a hacer, si no existe el servicio se crea para después darle presupuesto
Reporte de servicio	Documento	Servicios	Donde se genera la caída de información, formato físico o digital que aprueba un servicio iniciado

Figura 2.7 Descripción de las figuras de inicio de un nuevo servicio.

2.3.3 FIGURAS DE PROCESO PARA LAS FINANZAS.

Figura	Significado de la figura	Entidad	Ejecución
Se asigna un presupuesto	Proceso	Finanzas	Asigna un presupuesto de un servicio para llevar un control de ingresos, egresos y obtener ganancias.
Registra	Proceso dependiente de otro proceso	Finanzas	Registrar, aquí es donde se da de alta ingresos y egresos de un servicio.
Guarda los datos	Base de datos	Finanzas	Guardar datos, punto donde se guardar todas las acciones del presupuesto.
Obtiene ganancias	Proceso dependiente de otro proceso	Finanzas	Se calcula las ganancias de un servicio, restando los egresos de los ingresos.
Genera reporte con ingresos, egresos y ganancias	Proceso dependiente de otro proceso	Finanzas	Se genera el reporte de las ganancias con sus ingresos y egresos de un servicio.

Figura 2.8 Descripción de las figuras para finanzas.

2.3.4 REPORTE HECHO POR LA REPRESENTANTE LEGAL PARA EL DEPARTAMENTO DE FINANZAS.

Finanzas		
Ingreso	Por el concepto de enganche	
Egreso	por el concepto de . de Diesel	
Egreso	Por el concepto de 6 polines engrasados	
Egreso	Por el concepto de . de anticongelante	
Descripción Limpieza de un terreno		
Total de egresos:	, Total de ingresos:	a la fecha 15-08-2020

Figura 2.9 Registro de una nueva finanza para un servicio, forma de trabajo actual.

CAPITULO III: ELECCIÓN DE LA METODOLOGÍA.

3.1 ELECCIÓN DE LA METODOLOGÍA.

3.1.1 METODOLOGÍAS TRADICIONALES.

3.1.1.2 METODOLOGÍA TRADICIONAL DE CASCADA

Ventajas

- Esta metodología presenta varias ventajas, como la facilidad para la medición del progreso del proyecto o la posibilidad de que el cliente no se involucre mucho si así lo desea. Pero la mayor ventaja es su presupuesto cerrado, acordado con el proveedor desde el principio. Ya que no se harán cambios en el software en su proceso de desarrollo, el precio no cambiará.

Desventajas

- Como contrapunto, el no introducir ningún cambio en el desarrollo del software puede conllevar un mayor gasto a posteriori. Es posible que tras unos meses probando el programa los trabajadores echen en falta alguna funcionalidad sobre la que no se había pensado en el momento de definir las características que se pedían al desarrollador. Esto llevaría a solicitar un nuevo presupuesto para solucionar el problema, por lo que el dinero que en un principio parecía haberse ahorrado la compañía se ha de desembolsar ahora.

Diagrama

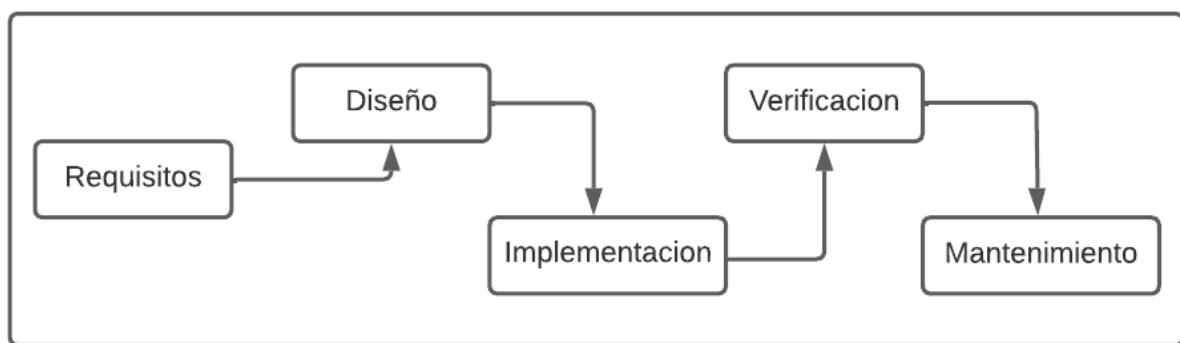


Figura 3.1 Diagrama de la metodología de cascada.

3.1.1.3 METODOLOGÍA TRADICIONAL INCREMENTAL

Ventajas

Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software.

- Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos.
- Es más fácil probar y depurar en una iteración más pequeña.
- Es más fácil gestionar riesgos.
- Cada iteración es un hito gestionado fácilmente

Diagrama

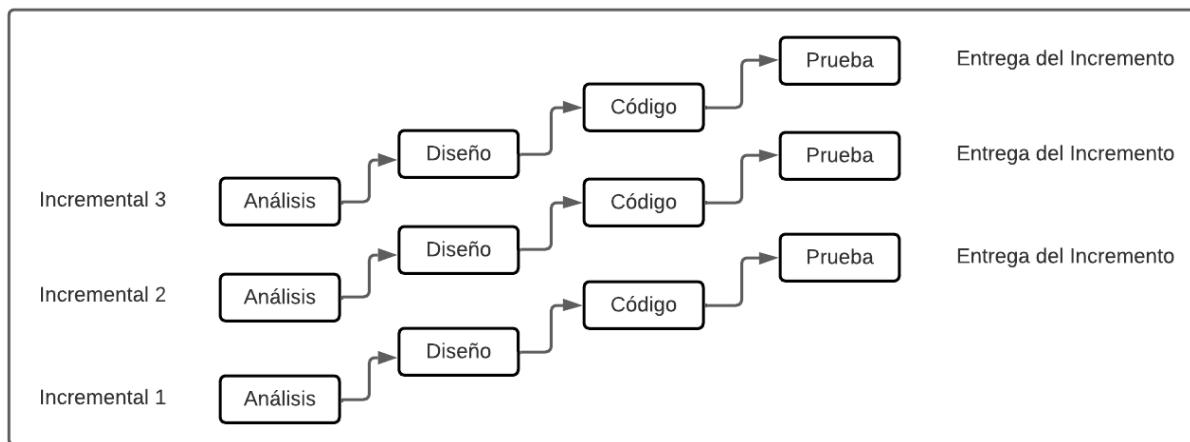


Figura 3.2 Diagrama de la metodología incremental.

3.1.1.4 METODOLOGÍA TRADICIONAL RUP

Ventajas:

- Un proceso de software hecho a la medida para ser publicado y hacerlo accesible para todo el equipo del proyecto. Esto quiere decir que cualquiera que se encuentre trabajando en el proyecto pueda acceder a este con más facilidad, evitando problemas relacionados a este tipo de cuestiones.

- Es una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo, pues los roles están muy bien definidos, y dictan quien realiza cada actividad, dependiendo del área en el que se desarrollara, de esta manera es bastante útil para definir roles en los proyectos.
- Reutilización. Los roles y distintos pueden ser reutilizados en proyectos futuros, dando como resultado una mejor organización al proyecto y menos utilización de recursos o tiempo, aspectos que se pueden emplear directamente en el proyecto.
- Mantenimiento más sencillo y modificaciones locales. Esta es una ventaja muy importante, pues si el proceso así lo permite es bastante más fácil poder realizar un cambio al proyecto en un futuro, sin generar perdidas o retrasos tan notorios o sobresalientes.
- Ofrece a cada usuario, un filtrado personalizado de la definición del proceso publicado, acorde con su rol dentro del proyecto.
- Es el proceso de desarrollo más general de los existentes actualmente. Es decir, este proceso es de los más utilizados para el desarrollo del software por la mayoría de las empresas, pues su enfoque es bastante óptimo y tiende a ser una metodología viable para la mayoría de estas.

Desventajas:

- Por el grado de complejidad puede ser no muy adecuado. Debido a que es un proceso bastante grande y complejo es muy común que no sea el adecuado para cualquier proyecto pequeño (cosa que se explicara en la siguiente desventaja), es por eso que a veces no puede ser el adecuado.
- En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios. Al ser una metodología bastante cara y con bastantes requerimientos en cuanto a roles (personales), a veces los costos son muy elevados, dando como resultado una imposibilidad por costear el proyecto.

- Método pesado. En muchos aspectos tiende a ser muy pesado, pues como se explicaba en los puntos anteriores, la complejidad es alta.

Diagrama

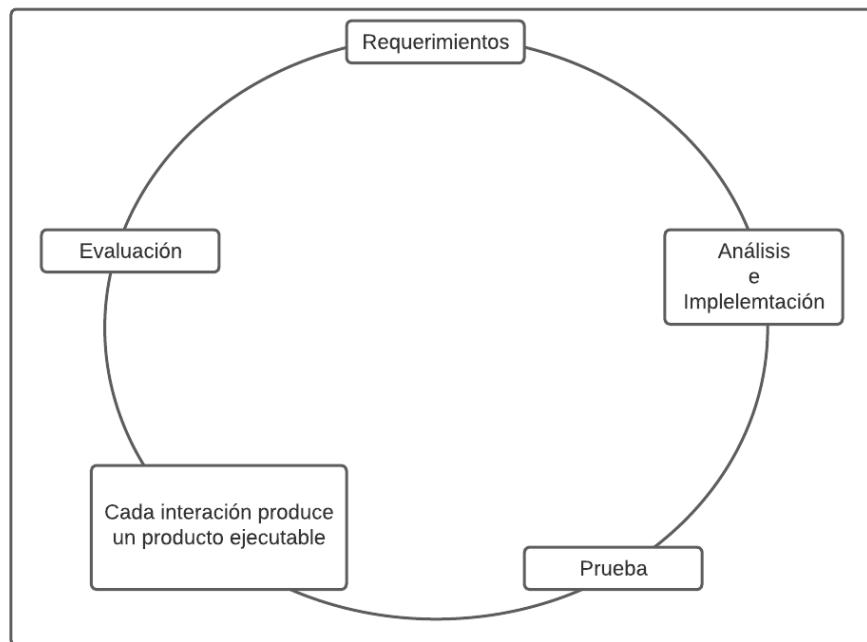


Figura 3.3 Diagrama de la metodología RUP.

3.1.2 METODOLOGÍAS AGILES

3.1.2.1 METODOLOGÍA ÁGIL SCRUM

Ventajas:

- Flexibilidad y adaptación a los contextos: Scrum puede adaptarse a cualquier área o sector de la gestión, es decir, no es una técnica exclusiva de ninguna disciplina.
- Gestión sistemática de riesgos: Trabajar con Scrum permite que los problemas que pueden contrarrestar los progresos un proyecto, pueden ser gestionados en el mismo momento de su aparición. Es decir, la intervención de los

equipos de trabajo puede ser inmediata cuando surjan conflictos o dificultades en el transcurso del proyecto.

- Sistema jerárquico de actividades: Plantear este sistema de actividades a ejecutar durante el ciclo de vida del proyecto, permite que los colaboradores puedan darle prioridad a aquello que se requiera con mayor urgencia frente a aquellas tareas que no son las más demandantes.
- Fechas realistas de entregas del proyecto: En Scrum, al trabajar con iteraciones, se segmenta el objetivo a entregar lo que hace que los márgenes de error sean menores como así también, que las entregas finales se ajusten a lo que fue planificado. Así, los Sprint o iteraciones hacen más fácil de gestionar las tareas y un mejor manejo de los tiempos.
- Feedbacks en el equipo: El trabajo de Scrum permite que el equipo establezca reuniones diarias donde pueden fijar qué se hizo, qué se hará y qué impedimentos que hay para realizarlo, pudiendo intercambiar opiniones e ideas acerca del proyecto.

Visión global: En Scrum, contar con una visión holística e integral del proyecto mientras está en curso es posible, además de que esto fortalece el equipo de trabajo y sus involucrados.

Desventajas:

Se aplica a equipos reducidos: Scrum es exitosa cuando se trabaja con grupos de pocos colaboradores. En una empresa grande, por ejemplo, se debe sectorizar o dividir en grupos que cuenten con objetivos concretos. De lo contrario, el efecto de la técnica no será el mismo.

- Requiere una exhaustiva definición de las tareas y sus plazos: Scrum funciona correctamente cuando tanto las tareas como el tiempo en que se ejecutarán cada una se encuentran definidos. La esencia de esta metodología reside en la división del trabajo de cada etapa y de sus tareas específicas.

- Difícil escalabilidad: Aplicar un enfoque Scrum para grandes proyectos se establece un reto ya que puede fallar la coordinación precisa, por lo que no garantiza que sea escalable a largo plazo.
- Puede necesitar de transformaciones dentro de la organización: En ocasiones, para trabajar con Scrum la empresa debe pasar por ciertas transformaciones organizativas en sus departamentos y áreas. Es la empresa quien debe gestionar y organizarse para que las colaboraciones sean exitosas.
- No se integra fácilmente con enfoque clásico de gestión de proyectos: El enfoque de Scrum no suele ser el adecuado para proyectos que requieren previsibilidad y un plan bien definido.

Diagrama

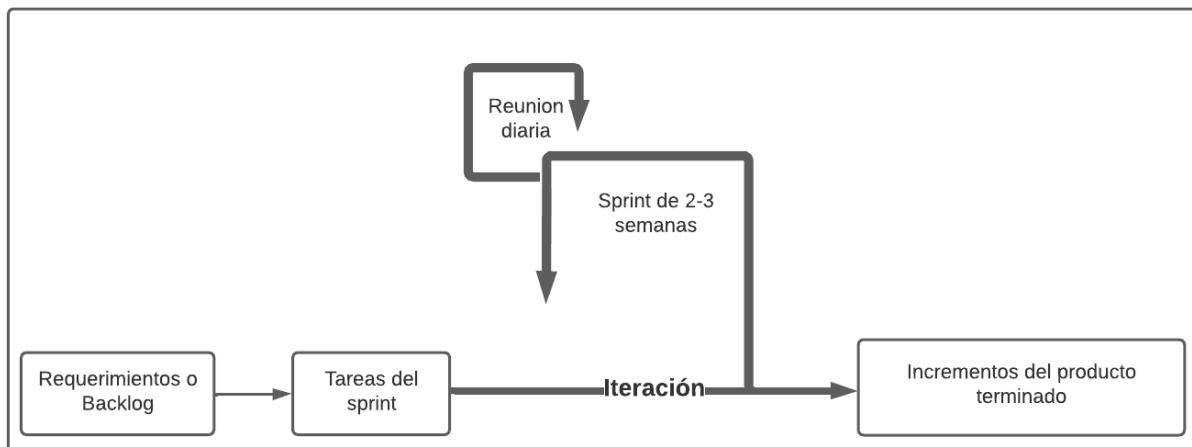


Figura 3.4 Diagrama de la metodología SCRUM.

3.1.2.2 METODOLOGÍA ÁGIL XP (EXTREME PROGRAMMING)

Ventajas:

- Da lugar a una programación sumamente organizada.
- Ocasiona eficiencias en el proceso de planificación y pruebas.
- Cuenta con una tasa de errores muy pequeña.
- Propicia la satisfacción del programador.
- Fomenta la comunicación entre los clientes y los desarrolladores.

- Permite ahorrar mucho tiempo y dinero.
- Puede ser aplicada a cualquier lenguaje de programación.
- El cliente tiene el control sobre las prioridades.
- Se hacen pruebas continuas durante el proyecto.
- La XP es mejor utilizada en la implementación de nuevas tecnologías.

Desventajas:

- Es recomendable emplearla solo en proyectos a corto plazo.
- En caso de fallar, las comisiones son muy altas.
- Requiere de un rígido ajuste a los principios de XP.

Diagrama

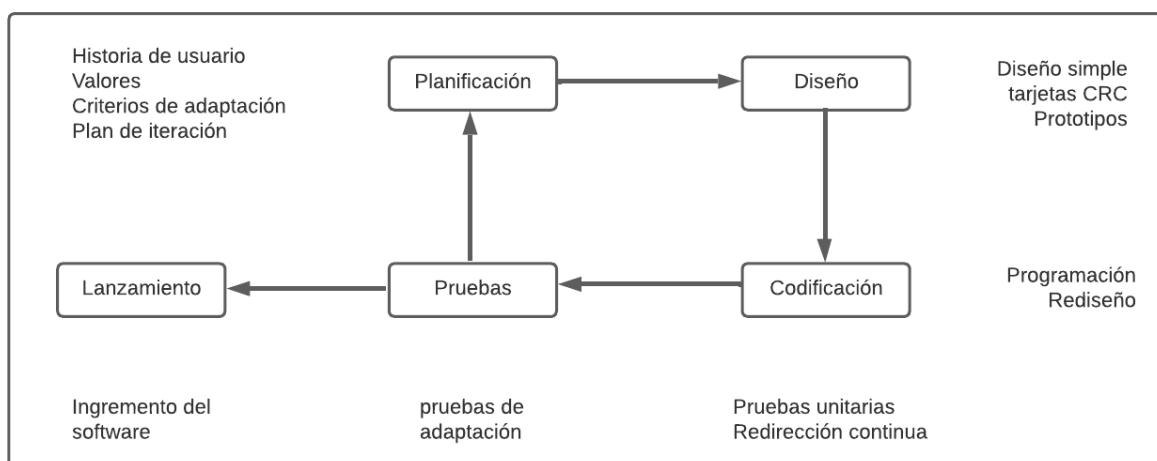


Figura 3.5 Diagrama de la metodología XP.

3.1.2.3 SCRUM MODIFICADO (MÉTODO AXOLOTL)

Ventajas

- Es una metodología ya usada por el equipo y que sumado a esto dio buenos resultados.
- Permite a principiantes valerse de herramientas para el correcto progreso del proyecto.

- Es excelente para proyectos que cuentan con un tiempo limitado de desarrollo
- El progreso y entrega del proyecto están directamente relacionados
- Permite que el cliente reciba exactamente lo que necesita

Desventajas

- Es un método completamente nuevo.
- Agregarlo a equipos de trabajo grandes requiere de seccionar en departamentos.
- Se requiere de una comunicación constante con el cliente o los implicados en el proyecto.

Diagrama

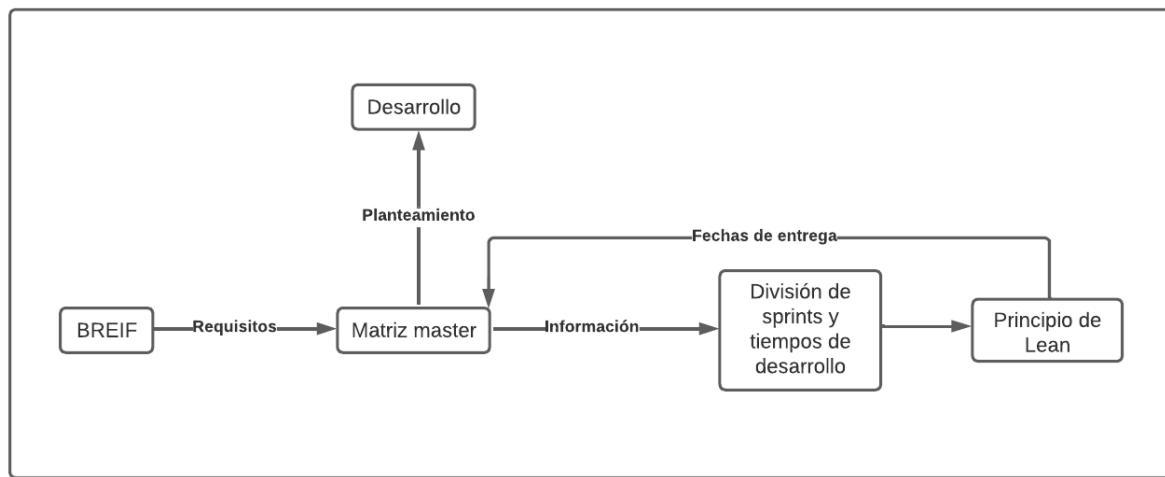


Figura 3.6 Diagrama de la metodología AXOTOLT.

3.2 ELECCIÓN DE LA METODOLOGÍA

¿Cual?

El método elegido fue el de método Axolotl.

¿Por qué?

Es una metodología que se ha usado previamente, que tuvo un buen desempeño y que acompañado de un software el cual permite de manera individual o grupal llevar a cabo el avance y organización del proyecto, cabe destacar que dicho software fue creado por los mismos desarrolladores del método con el fin de facilitar el desarrollo.

Caso de éxito.

A través de la consultoría de software con el nombre “Anaconda Software S.A. de C.V.” se desarrolló un sistema de tipo administrativo para el “Instituto de Capacitación de la Industria de la Construcción (ICIC)” organiza de manera interna a la empresa en sus egresos, ingresos, usuarios, empleados, clientes, proveedores, contabilidad y administración, todo esto a nivel nacional en las 32 delegaciones que lo conforman en la república mexicana; dando orden a más de 480 empleados, 50 proveedores y una base de 14800 clientes que se incrementa día a día. Debido a esto se optó por utilizar una metodología ágil que partiera de las bases de la metodología modificándola para hacer el método Axolotl, cumpliendo de esta manera con el proyecto.

CAPITULO IV: APLICACIÓN DE LA METODOLOGÍA SCRUM MODIFICADO (MÉTODO AXOLOTL)

4.1 APLICACIÓN DE LA METODOLOGÍA

4.1.1 BREIF.

Nombre del proyecto

Constructora Matzy.

Antecedentes

La constructora Matzy es una empresa dedicada a la construcción a través de máquinas de construcción y tractores de carga, las cuales son alquiladas por empresas u organismos externos, teniendo inventarios, gastos, cómputo de horas de trabajo, vehículos varios, etc.

Necesidades

Para aclarar mejor el problema, el cliente y el Project Manager pueden establecer necesidades básicas de diseño, desarrollo, información, etc.

- Conteo de horas de trabajo
- Elaboración de reportes internos para servicio
- Elaboración de reportes internos para finanzas
- Notificaciones
- Catálogos de vehículos
- Organización de tiempos y fechas
- Accesibilidad al software desde cualquier lugar

Categoría

A qué tipo de industria pertenece. Construcción y es una sociedad de capital variable.

Tipo

Administrativo interno (Software para organizar sus funciones).

Productos o servicios

En este caso será un sistema interno entonces lo que pretende administrar es sus ingresos, egresos, vehículos, mercancías, horas de trabajo, etc.

4.2 REQUISITOS.

Requiere de un software de tipo web que le permita acceder en cualquier lugar con solo acceso a internet, a continuación, se listan los requisitos para la elaboración del software:

- Gestionar usuarios para el control de ingreso de datos.
- Módulo dispuesto a administrar sus finanzas y reportes para las mismas.
- Modulo que facilite el inicio de un servicio y la toma de datos del cliente para una posterior asignación de presupuesto que a su vez le genere el reporte para tenerlo física y digitalmente.
- Catalogo con sus vehículos que permita saber la disponibilidad de los mismos y si estos no están disponibles entonces se le asignara una fecha con disponibilidad.
- Gestión de fechas y horas de trabajo para los vehículos.

4.3 PRODUCT BACKLOG.

ID	Enunciado de la historia
H1	Yo, "representante legal de la Constructora Matzy ", necesito de un "modulo" de seguridad, con la finalidad de que solo yo pueda acceder a la información en el software y un "modulo" que me permita tener datos resumidos del sistema en general, con la finalidad de tener dicha información al entrar al software.
H2	Yo, "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita darles acceso a nuevos usuarios, con la finalidad de que mi personal administrativo pueda acceder a la información y/o crear nueva, una vez registrados.

H3	Yo, "representante legal de Constructora Matzy", necesito un "módulo" que me permita recabar los datos necesarios del cliente y verificar la disponibilidad de mis vehículos y las fechas en que estarán disponibles, y de esta manera, puedo decidir correctamente cuándo iniciar un servicio, obteniendo un informe interno con todo lo registrado y a la vez uno que me permita registrar mis vehículos, con el fin de registrar el vehículo una sola vez y no cada nuevo servicio.
H4	Yo, "representante legal de Constructora Matzy", necesito un "módulo" que me permita asignar un presupuesto a un servicio que ya se encuentra iniciado, para poder registrar mis ingresos y gastos correspondientes a este servicio con el cliente de turno y así obtener las ganancias, obteniendo un informe interno con todo lo registrado.

Id	Alias	Estado	Esfuerzo	Sprint	Prioridad
H1	Perfiles	Desarrollo	Medio	Sprint 1	Alta
H2	Personal	Desarrollo	Bajo	Sprint 2	Media
H3	Departamento de servicios	Desarrollo	Alto	Sprint 3	Alta
H4	Departamento de finanzas	Desarrollo	Alto	Sprint 4	Alta

4.4 TABLERO DE RESPONSABILIDADES.

Nombre	Roles	Responsabilidades
Lic. Mónica Guerrero Pérez	Dueño de la empresa	Revisión y aprobación final de Sprint
Raul Guerrero Ramírez	Líder de proyecto, Scrum Master, UX/UI	Levantamiento de requisitos, maquetación del sistema y testeo.
Antonio Juárez Andrade	Equipo de trabajo, desarrollador	Programación de interfaces y programación lógica del sistema

4.5 DESARROLLO.

SPRINT BACKLOG SP1 H1.

ID	Product backlog		
H1	Yo, "representante legal de la Constructora Matzy ", necesito de un "modulo" de seguridad, con la finalidad de que solo yo pueda acceder a la información en el software y un "modulo" que me permita tener datos resumidos del sistema en general, con la finalidad de tener dicha información al entrar al software.		
ID	Iteración		
H1	Sprint 1		
ID	Tarea	Voluntario	Total, de Hrs.
T1	Preparativos y Diseño UX	Raul Guerrero Ramírez	20Hrs.
T2	Diseño UX del Menú general y pagina con la que el usuario se encontrara al acceder al sistema, respetando un entorno actual para el usuario el cual le implique una curva de aprendizaje corta.	Raul Guerrero Ramírez	20Hrs.

T3	Definición de los tres principios del UX para el software y su experiencia de usuario.	Raul Guerrero Ramírez	20Hrs.
T4	Análisis de la imagen y cromática de la empresa	Raul Guerrero Ramírez	20Hrs.
T5	Maquetación del módulo de ingreso y Pagina de ingreso al sistema		20Hrs.
T6	Levantamiento del sistema mediante framework de back-end y creación de base de datos.	Antonio Juarez Andrade	20Hrs.
T7	Instalación de framework front-end Bootstrap y AdminLTE (plantilla), será la base de todo el diseño del proyecto.	Antonio Juarez Andrade	20Hrs.
T8	Maquetación programada, programación de la vista de perfiles junto con el diseño general del sistema y vista principal (home).	Antonio Juarez Andrade	20Hrs.

Horas repartidas entre los días de desarrollo del sprint del 1 al 20																				
ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T1	5	5	5	5																
T2					5	5	5	5												
T3								5	5	5	5									
T4												5	5	5	5					
T5																5	5	5	5	
T6									5	5	5	5								
T7											5	5	5	5						
T8																5	5	5	5	

TABLERO DE TAREAS INICIO DEL SP1 H1.

Historia de usuario	Tareas	En proceso	Concluido
H1 Perfiles	SP1 H1 T1	T1	
H1 Perfiles	SP1 H1 T2	T2	
H1 Perfiles	SP1 H1 T3	T3	
H1 Perfiles	SP1 H1 T4	T4	
H1 Perfiles	SP1 H1 T5	T5	
H1 Perfiles	SP1 H1 T6	T6	
H1 Perfiles	SP1 H1 T7	T7	
H1 Perfiles	SP1 H1 T8	T8	
Porcentaje de avance			
0-20%	21-40%	41-60%	61-80%
T1,T2,T3,T4,T5,T6,T7,T8			

PRINCIPIO DE LEAN SP1 H1

El desarrollo de este sprint esta estrictamente sujeto a lo requerido en la historia de usuario, lo cual es controlar el acceso individual al sistema y contar con un módulo que muestre un resumen de los registros y así, enfocando el desarrollo en la demanda y no en la oferta. Mediante este principio se evita la creación de funciones innecesaria que el usuario nunca usara y se agiliza los tiempos de desarrollo.

SP1 H1 T6 LEVANTAMIENTO DEL SISTEMA.

Para el levantamiento del sistema se utilizará un esquema de trabajo o también llamado framework, se usará Laravel que es un framework back-end que facilita seguir el

estilo de arquitectura de software (MVC), este framework estará acompañado también de un kit de inicio Jetstream para la implementación de inicio de sesión.

Para probar o tener host local se usará Laragon que trabaja muy bien con Laravel, es una herramienta de desarrollo web muy útil, por medio de esta herramienta podremos instalar los frameworks que serán usados para el proyecto.

Instalación de Laravel mediante terminal “Cmder” de Laragon.

```
C:\laragon\www>laravel new matzy_system
[laravel]
```

Figura 4.1 Instalación de framework Laravel por comandos mediante Cmder.

Instalación de Jetstream mediante terminal “Cmder” de Laragon.

```
Laravel Mix v6.0.49
✓ Compiled Successfully in 3437ms      php artisan jetstream:install inertia
File          Size
/js/app.js    696 Kib
/css/app.css  53.3 Kib
webpack compiled successfully      La pila de inercia también se puede instalar con soporte SSR:
C:\laragon\www\matzy_system>
```

Figura 4.2 Instalación de kit de inicio de sesión Jetstream por comandos mediante Cmder.

Para la creación de la base de datos será usado phpMyAdmin como interface para el control de la base de datos, el nombre será “matzy_system”.

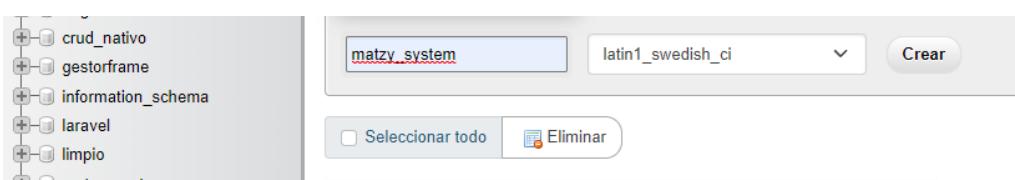


Figura 4.3 Creación de la base de datos mediante el gestor phpMyAdmin.

El esquema de la base de datos es el siguiente.

The screenshot shows a database schema diagram with the following tables:

- matzy_system failed_jobs**: Contains columns: id (bigint(20) unsigned), uuid (varchar(255)), connection (text), queue (text), payload (longtext), exception (longtext), and failed_at (timestamp).
- matzy_system password_resets**: Contains columns: email (varchar(255)), token (varchar(255)), and created_at (timestamp).
- matzy_system sessions**: Contains columns: id (varchar(255)), user_id (bigint(20) unsigned), ip_address (varchar(45)), user_agent (text), payload (text), and last_activity (int(11)).
- matzy_system users**: Contains columns: id (bigint(20) unsigned), name (varchar(255)), ape_pat (varchar(255)), ape_mat (varchar(255)), foto (varchar(255)), edad (int(11)), dirección (varchar(255)), tipo_user (int(11)), email (varchar(255)), email_verified_at (timestamp), password (varchar(255)), two_factor_secret (text), two_factor_recovery_codes (text), two_factor_confirmed_at (timestamp), remember_token (varchar(100)), current_team_id (bigint(20) unsigned), profile_photo_path (varchar(2048)), created_at (timestamp), and updated_at (timestamp).
- matzy_system personal_access_tokens**: Contains columns: id (bigint(20) unsigned), tokenable_type (varchar(255)), tokenable_id (bigint(20) unsigned), name (varchar(255)), token (varchar(64)), abilities (text), last_used_at (timestamp), created_at (timestamp), and updated_at (timestamp).

Figura 4.4 Tablas para el inicio de sesión que usara Jetstream, se crean al instalar el kit al proyecto.

Las tablas encerradas con la línea roja son propias del kit de inicio de sesión de Jetstream, la tabla es una de las más importantes ya que es donde se almacenarán todos los usuarios que existirán en el sistema, la tabla “password_resets” es para implementar la recuperación de contraseña, “sessions” es para tener control de las sesiones activas de un usuario en distintos dispositivos.

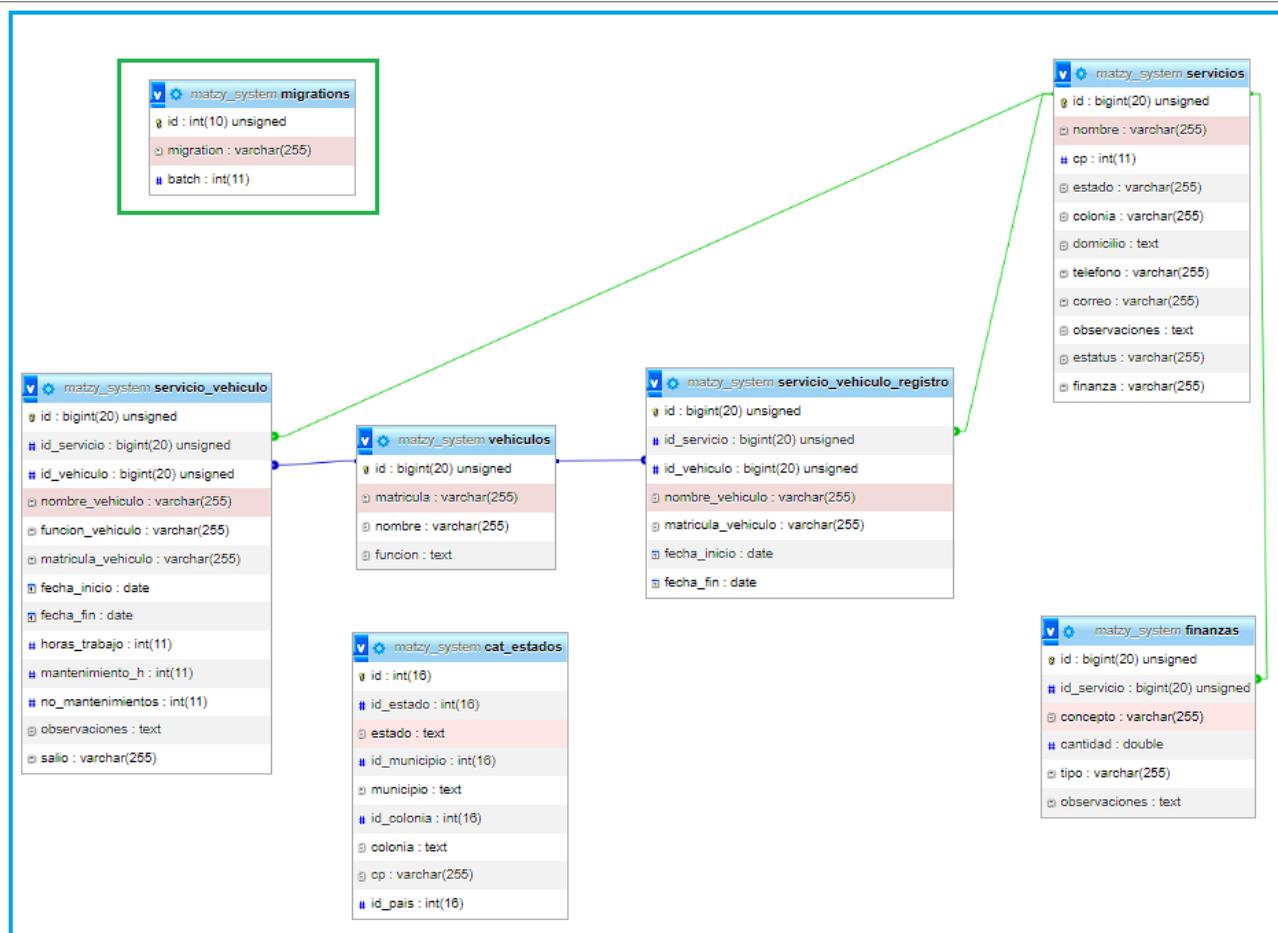


Figura 4.5 Tablas que almacenaran los datos del flujo de trabajo de la empresa.

Las tablas encerradas por la línea verde son propias de Laravel para llevar un control de las tablas agregadas en el sistema.

Las tablas encerradas por la línea azul son para el control del funcionamiento en general del sistema y para el modelo de negocios de la empresa. La tabla “servicios” es la tabla principal de todo el sistema y es donde se agregarán los nuevos servicios, las tablas “servicio_vehiculo” y “servicio_vehiculo_registro” comparten muchos campos y estas son para llevar el control de los vehículos agregados para un servicio así, como saber que vehículos están disponibles. La tabla “vehículos” es donde se agregarán todos los vehículos de la empresa, la tabla “finanzas” es donde se llevará los registros de los ingresos y egresos de un servicio, así como sus ganancias y la tabla “cat_estados” es

donde estarán todas las localidades según el código postal (cp.) esta es una tabla extra fuera del flujo de trabajo de la empresa.

Diagramas de entidad-relación (ER) de las tablas principales del sistema.

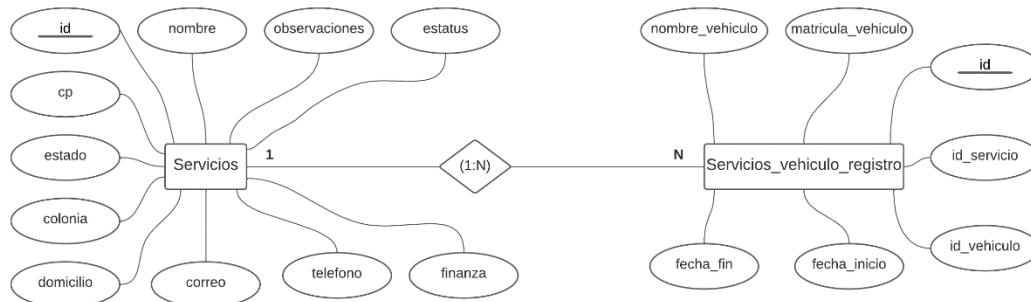


Figura 4.6 Diagrama (ER) de las tablas “servicios” y “servicios_vehiculo_registro” con carnalidad de uno a muchos.

La tabla “servicios” tiene una carnalidad de uno a muchos (1:N) con la tabla “servicios_vehiculo_registro” por qué un registro de “servicios” puede tener muchos registros de “servicios_vehiculo_registro” ya que esta conexión indica que vehículos terminaron su tiempo de servicio.

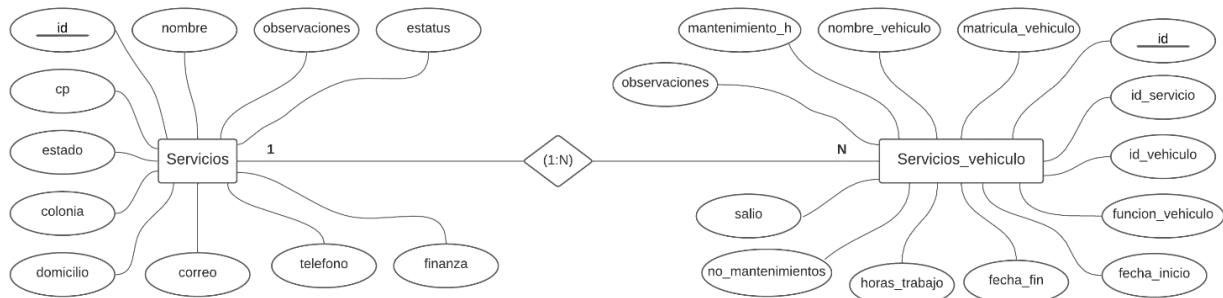


Figura 4.7 Diagrama (ER) de las tablas “servicios” y “servicios_vehiculo” con carnalidad de uno a muchos.

La tabla “servicios” tiene una carnalidad de uno a muchos (1:N) con la tabla “servicios_vehiculo” por qué un registro de “servicios” puede tener muchos registros de “servicios_vehiculo” ya que esta conexión indica que vehículos fueron registros con el servicio y así tener un control de los mismos, un vehículo no puede existir en dos servicios al mismo tiempo.

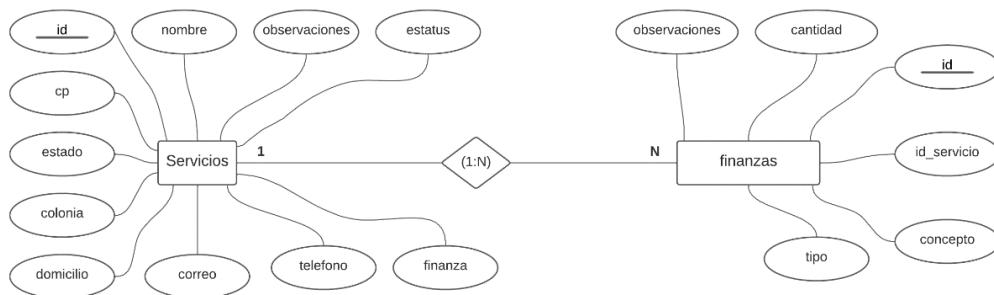


Figura 4.8 Diagrama (ER) de las tablas “servicios” y “finanzas” con carnalidad de uno a muchos.

La tabla “servicios” tiene una carnalidad de uno a muchos con la tabla “finanzas” ya que uno servicio puede tener muchas finanzas, pero no al viceversa, esta conexión es para saber los ingresos y egresos de un servicio con el fin de poder tener una ganancia.

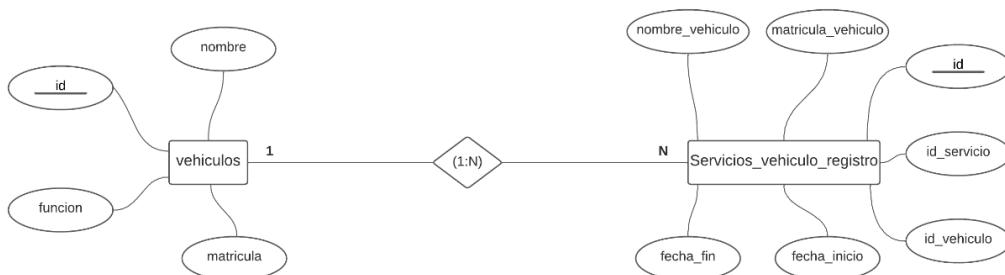


Figura 4.9 Diagrama (ER) de las tablas “vehículos” y “servicios_vehiculo_registro” con carnalidad de uno a muchos.

La tabla “vehículos” tiene una carnalidad de uno a muchos (1:N) con la tabla “servicios_vehiculo_registro” ya que esta conexión sirve para saber que vehículos terminaron su tiempo de trabajo en dicho servicio y así poder filtrar los que ya pueden ser usados en otro servicio.

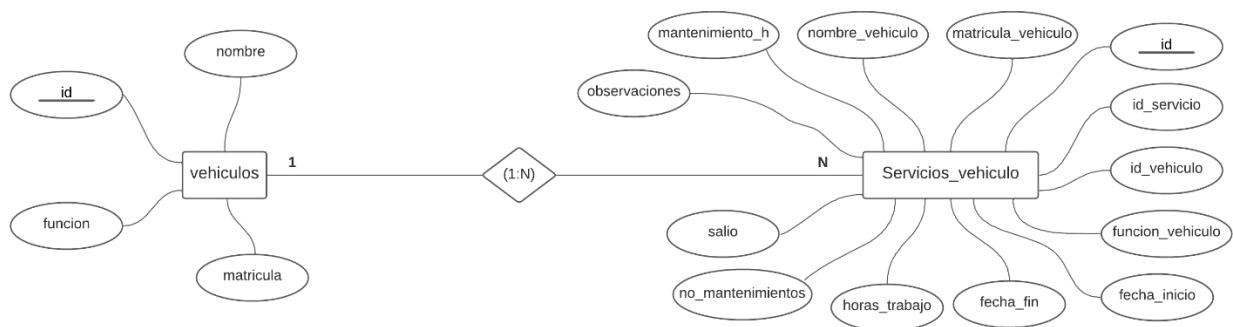


Figura 4.10 Diagrama (ER) de las tablas “vehículos” y “servicios_vehiculo” con carnalidad de uno a muchos.

La tabla “vehículos” tiene una carnalidad de uno a muchos (1:N) con la tabla “servicios_vehiculo” ya que esta conexión sirve para saber que vehículos se agregaron en el servicio y así tener un control del uso de los mismos.

El sistema hasta este punto.

The image shows a login interface. At the top center is a blue circular logo. Below it is a white rectangular form. The form contains fields for 'Email' and 'Password', each with an input box. Below these is a 'Remember me' checkbox. At the bottom right of the form are two buttons: a light blue button labeled 'Forgot your password?' and a dark blue button labeled 'LOG IN'.

Figura 4.11 Interfaz de inicio de sesión del sistema.

SP1 H1 T7 INSTALACIÓN DE FRAMEWORKS FRONT-END.

Para el diseño también existen muchos frameworks que ayudan a recortar los tiempos de diseño programado, para ello usaremos Bootstrap y este estará acompañado de una plantilla que usa dicho framework que es AdminLTE.

Instalación de Bootstrap mediante terminal “Cmder” de Laragon.

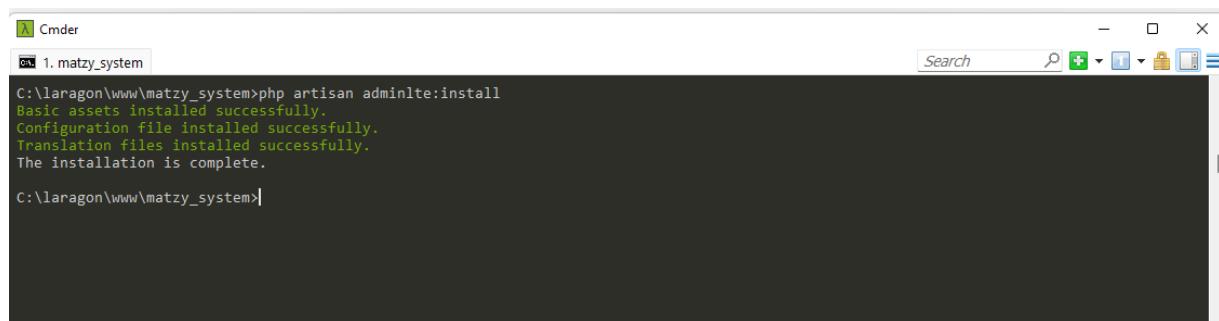


```
Laravel Mix v6.0.49
  - ex-coopasec.yaml archivo, actualice los puertos HMR ({$HMR_PORT:-5173})/5173 o
    su puerto que su servidor vite dev esté sirviendo a sus activos.

  ✓ Compiled Successfully in 7476ms
  ┌─────────────────────────────────────────────────────────────────────────────────┐
  │ build:                                File   Size
  │ └─── dockerfile, Dockerfile           /js/app.js  2.46 MiB
  │      args:                            css/app.css  230 Kib
  └─────────────────────────────────────────────────────────────────────────┘
webpack compiled successfully
mail-0.3/app
C:\laragon\www\matzy_system>host.docker.internal host-gateway
```

Figura 4.12 Instalación del framework de front-end Bootstrap mediante Cmder.

Instalación de AdminLTE mediante terminal “Cmder” de Laragon.



```
Cmder
1. matzy_system
C:\laragon\www\matzy_system>php artisan adminlte:install
Basic assets installed successfully.
Configuration file installed successfully.
Translation files installed successfully.
The installation is complete.

C:\laragon\www\matzy_system>
```

Figura 4.13 Instalación de la plantilla AdminLTE mediante Cmder.

Para agregar la plantilla de AdminLTE se agrega las siguientes directivas, @extends le permite a Laravel "extender" una plantilla, que define sus propias secciones, estilos y código JavaScript que la misma plantilla requiera por eso se agregan dichas sentencias y así se ahorra mucho código HTML.

```

C:\laragon\www\matzy_system\resources\views\vista.blade.php (matzy_system) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
matzy_system
  app
  bootstrap
  config
  database
    factories
    migrations
    seeders
    .gitignore
  node_modules
  public
  resources
    js
    lang
    markdown
    sass
  views
    api
    auth
    layouts
    profile
    vendor
      dashboard.blade.php
      navigation-menu.blade.php
      policy.blade.php
      terms.blade.php
    vista.blade.php
    welcome.blade.php
routes
  api.php
  channels.php
  console.php
  web.php
storage
tests
vendor
.editorconfig
.env
.env.example
.gitattributes
.gitignore
.styleci.yml
artisan
composer.json
composer.lock
package-lock.json
package.json

vista.blade.php
1 @extends('adminlte::page')
2
3 @section('title', 'Dashboard')
4
5 @section('content_header')
6   <h1>Dashboard</h1>
7 @stop
8
9 @section('content')
10   <p>Welcome to this beautiful admin panel.</p>
11 @stop
12
13 @section('css')
14   <link rel="stylesheet" href="/css/admin_custom.css">
15 @stop
16
17 @section('js')
18   <script> console.log('Hi!'); </script>
19 @stop

```

Figura 4.14 Código de las directivas de la plantilla AdminLTE para su implementación en las vistas.

SP1 H1 T8 MAQUETACIÓN PROGRAMADA.

Para empezar a dar el diseño a la vista inicio de sesión, se agregó CSS en algunas etiquetas del HTML, para cargar una foto de fondo se agregó por medio del atributo “style”.



```

1 <!DOCTYPE html>
2
3 <html lang="{{ str_replace('_', '-', app()->getLocale()) }}" style="background-image: url(./imagenes/Fondo.jpg);
4   background-size: cover;
5   background-repeat: no-repeat;
6   background-position: center center; background-attachment: fixed; ">
7

```

Figura 4.15 Estructura para darle diseño al fondo de la vista de inicio de sesión.

Después en la parte inferior se agregó más CSS para indicar que el contenedor fuera transparente y de igual forma estas reglas de CSS fueron agregadas mediante el atributo “style”.



```

25 <body class="font-sans antialiased" style="background-color: rgba(0, 0, 0, 0);">
26   {{ $slot }}
27 </body>
28 </html>

```

Figura 4.16 Código para darle transparencia a la vista de inicio de sesión.

Para el formulario se agregó algunos estilos para que tenga mejor vista.



```

1 <x-guest-layout>
2   <x-jet-authentication-card>
3     <x-slot name="logo">
4       <x-jet-authentication-card-logo />
5     </x-slot>
6
7   <div class="container-fluid fixed-top" style="background-color: #000000; height: 12%; padding-right: 0px;
8     padding-left: 0px;">
9
10    
11
12    <div class="col-12">
13      <div style="position: absolute; text-align: right; margin-top: -50px; padding-right: 15px; width: 100%;">
14        @if (Route::has('login'))
15          <div class="">
16            @auth
17              <a href="{{ url('Home') }}" class="btn" style="color: black; background-color: #F5B32B;">
18                Inicio</a>
19              @endif
20            </div>
21          @endif
22        </div>
23      <div style="width: 100%; background-color: #F5B32B; height: 15px; bottom: 0px;"></div>
24
25 </div>

```

Figura 4.17 Estructura XML/HTML para la vista de inicio de sesión del sistema.

```

72 <script type="text/javascript">
73
74     function activar_button(){
75         if (document.getElementById("correo").value!="" && document.getElementById("contrasena").value!=""){
76             document.getElementById("butt").style.backgroundColor= "#F5B32B";
77             document.getElementById("butt").disabled = false;
78         }else{
79             document.getElementById("butt").style.backgroundColor= "rgba(0, 0, 0, 0)";
80             document.getElementById("butt").disabled = true;
81         }
82     }
83 </script>

```

master (1) Spaces: 4 PHP

Figura 4.18 Función de JavaScript para activar o desactivar el botón de ingresar al sistema.

En la parte inferior se agregó código JavaScript, se creó la función “activar_button()” lo que hace esta función es verificar si ya fueron llenados los campos, si esto es cierto entonces activamos el botón de envío de formulario.

Con esto tenemos el siguiente resultado.

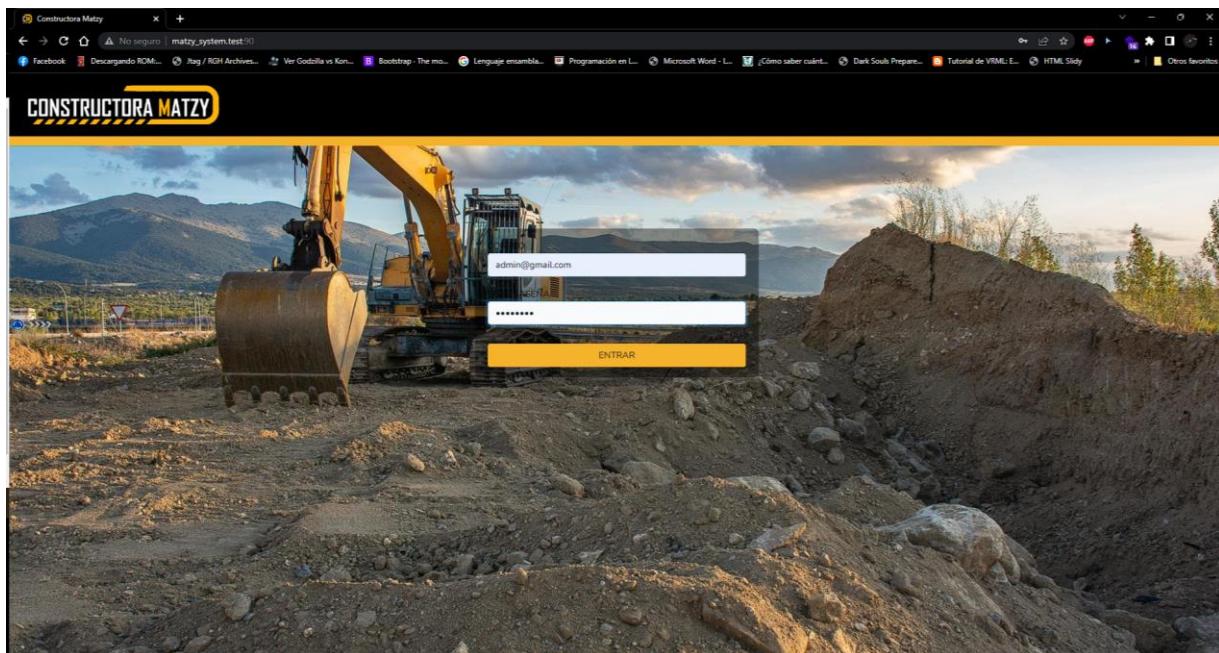


Figura 4.19 Vista final de inicio de sesión del sistema, diseño apegado a los estándares actuales.

Para el diseño base que tendrá el sistema en general, se cambia algunas variables del archivo de configuración de AdminLTE, permite editar todo el estilo de la plantilla.

```

56     | Admin Panel Logo
57     |-----
58     |
59     | Here you can change the Logo of your admin panel.
60     |
61     | For detailed instructions you can look the Logo section here:
62     | https://github.com/jeroennoten/Laravel-AdminLTE/wiki/
63     | Basic-Configuration
64     |
65     */
66
67     'logo' => "MATZY SYSTEM",
68     'logo_img' => 'imagenes/logo3p.png',
69     'logo_img_class' => ' brand-image img-circle elevation-3',
70     'logo_img_xl' => 'imagenes/logonuevoB.png',
71     'logo_img_xl_class' => 'brand-image-xl',
72     'logo_img_alt' => 'MATZY',
73
74     /*

```

Figura 4.20 Código de configuraciones de la plantilla AdminLTE, títulos y logos.

Aquí se cambia por los logos de la empresa y dejando las clases de CSS por defecto.

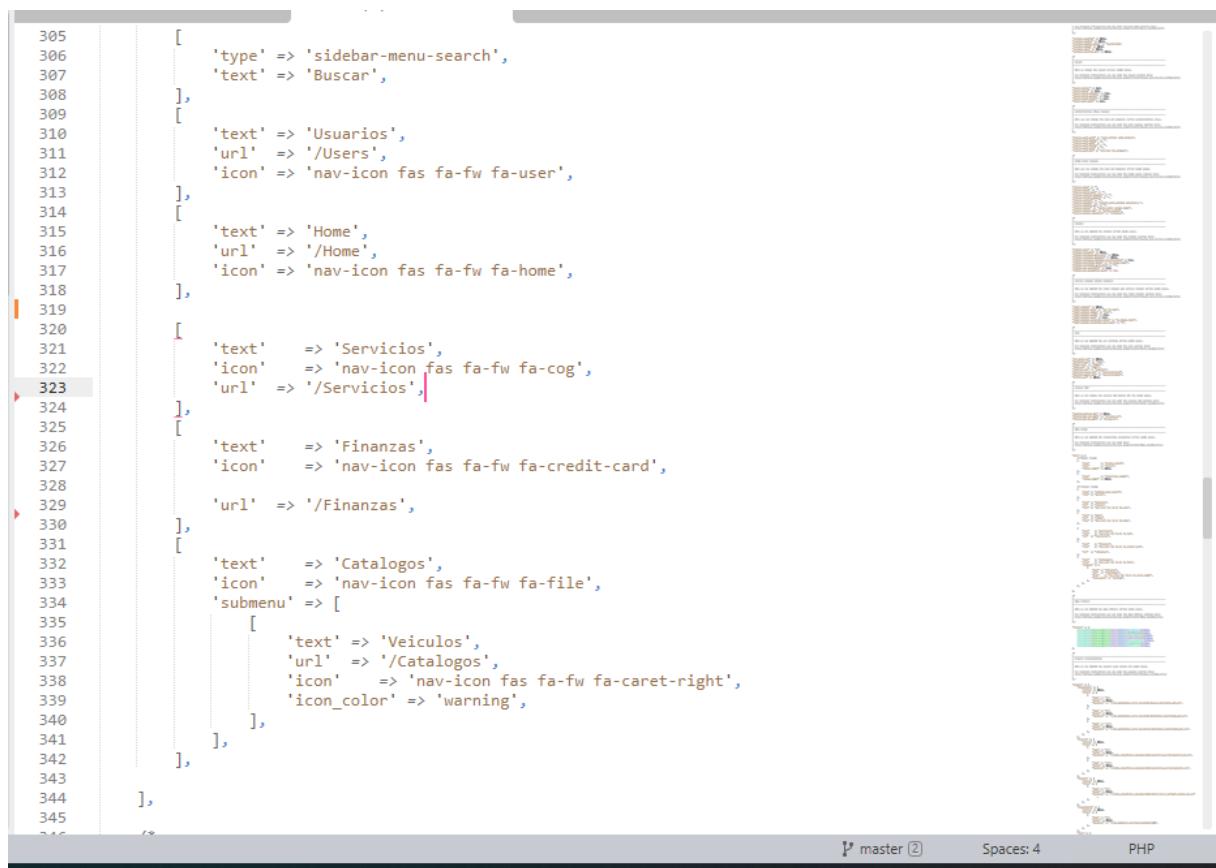
```

'classes_body' => '',
'classes_brand' => '',
'classes_brand_text' => '',
'classes_content_wrapper' => '',
'classes_content_header' => '',
'classes_content' => '',
'classes_sidebar' => 'sidebar-dark-warning elevation-4',
'classes_sidebar_nav' => '',
'classes_topnav' => 'navbar-white navbar-light',
'classes_topnav_nav' => 'navbar-expand',
'classes_topnav_container' => 'container',

```

Figura 4.21 Código de configuraciones de la plantilla AdminLTE, estilos y colores.

Aquí es la parte más importante, se indica el estilo de toda la plantilla, se usa la clase predefinida de AdminLTE “sider-dark-warning” que indica fondo negro y selección amarilla y “navbar-light” para el fondo del contenido blanco.



```

305      [
306        'type' => 'sidebar-menu-search',
307        'text' => 'Buscar',
308      ],
309      [
310        'text' => 'Usuarios',
311        'url' => '/Users',
312        'icon' => 'nav-icon fas fa-fw fa-user',
313      ],
314      [
315        'text' => 'Home',
316        'url' => '/Home',
317        'icon' => 'nav-icon fas fa-fw fa-home',
318      ],
319      [
320        'text' => 'Servicios',
321        'icon' => 'nav-icon fas fa-fw fa-cog',
322        'url' => '/Servicios',
323      ],
324      [
325        'text' => 'Finanzas',
326        'icon' => 'nav-icon fas fa-fw fa-credit-card',
327        'url' => '/Finanzas',
328      ],
329      [
330        'text' => 'Catalogos',
331        'icon' => 'nav-icon fas fa-fw fa-file',
332        'submenu' => [
333          [
334            'text' => 'Veiculos',
335            'url' => '/Catalogos',
336            'icon' => 'nav-icon fas fa-fw fa-caret-right',
337            'icon_color' => 'warning',
338          ],
339        ],
340      ],
341    ],
342  ],
343],
344],
345]
  
```

Figura 4.22 Código de configuraciones de la plantilla AdminLTE, apartados y enlaces.

En esta parte se agregarán los apartados o módulos que tendrá el sistema, 5 apartados son los que tendremos en todo el sistema.

También serán creados los controladores de cada uno de los apartados creados anteriormente para empezar a seguir la arquitectura de software MVC.

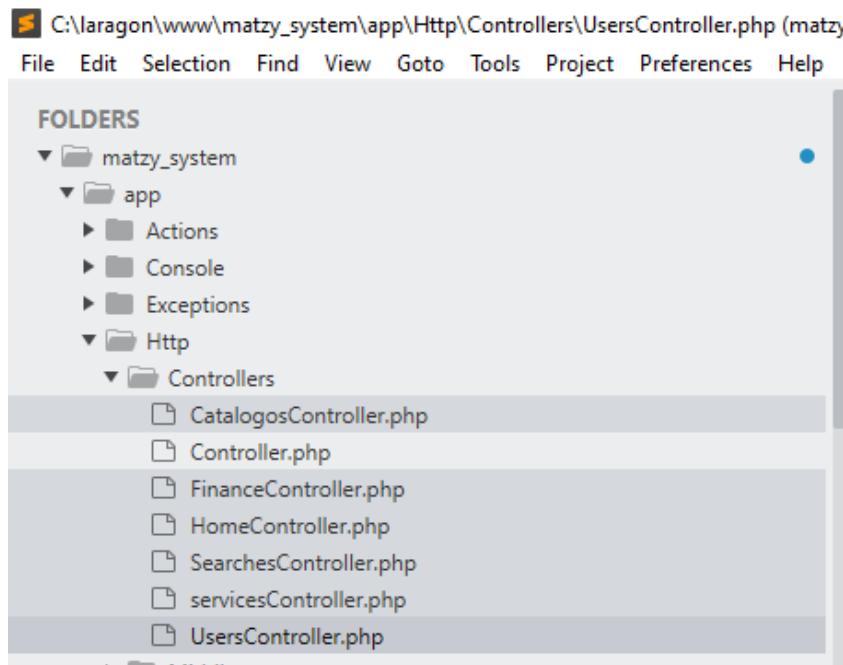


Figura 4.23 Creación de los controladores que serán usados para el flujo del sistema.

Las carpetas con las vistas de cada uno de los controladores.

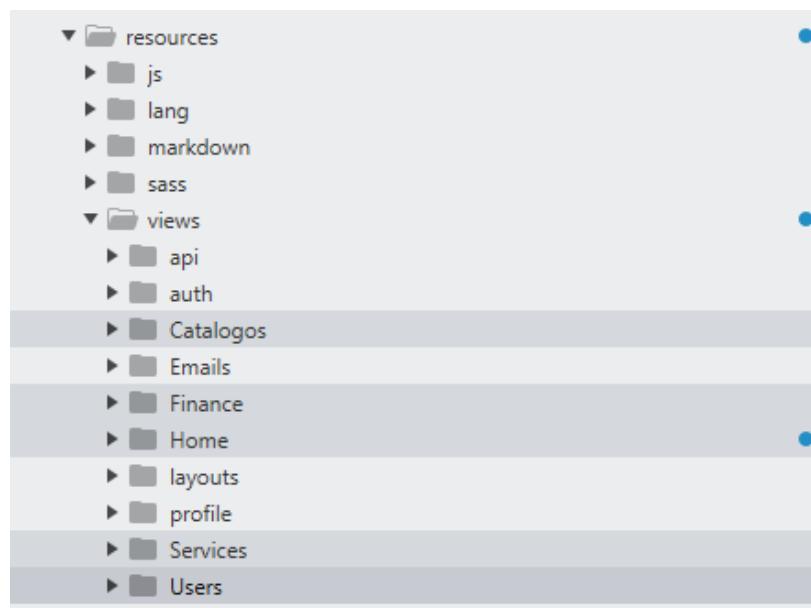


Figura 4.24 Creación de las carpetas donde estarán las vistas de cada uno de los apartados o módulos.

Para la construcción de la vista home será usado el controlador “HomeController” que será el que retornará dicha vista.

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use DB;
7 class HomeController extends Controller
8 {
9     public function __construct(){
10         $this->middleware('auth');
11     }
12
13     public function view_home(){
14
15         return view('Home.home');
16     }
17 }
18
19 }
```



Figura 4.25 Código del Controlador “HomeController”.

La vista “home” tiene lo siguiente.

```

35 <div class="card">
36     <div class="card-body">
37
38         <div class="row">
39
40             <!-- vehiculos disponibles -->
41             <div class="col-md-6"> ...
42
43             </div>
44             <!-- vehiculos con fecha terminada -->
45             <div class="col-md-6"> ...
46
47             </div>
48
49             <!--servicios ultimo -->
50             <div class="col-md-6"> ...
51
52             </div>
53
54             <!--servicios con finanza -->
55             <div class="col-md-6"> ...
56
57             </div>
58
59             <!--ultimo ingreso -->
60             <div class="col-md-6"> ...
61
62             </div>
63
64             <!--ultimo egreso -->
65             <div class="col-md-6"> ...
66
67             </div>
68
69         </div>
70     </div>
71 
```



Figura 4.26 Código resumido de la estructura HTML de la vista “Home”.

Prácticamente cada sección será igual solo con la diferencia que se manejaran diferentes colores y títulos que lo hará distinto uno del otro, para el diseño de cada sección se ocuparon algunas arquitecturas predefinidas por AdminLTE, solo agregamos iconos y algunos títulos.

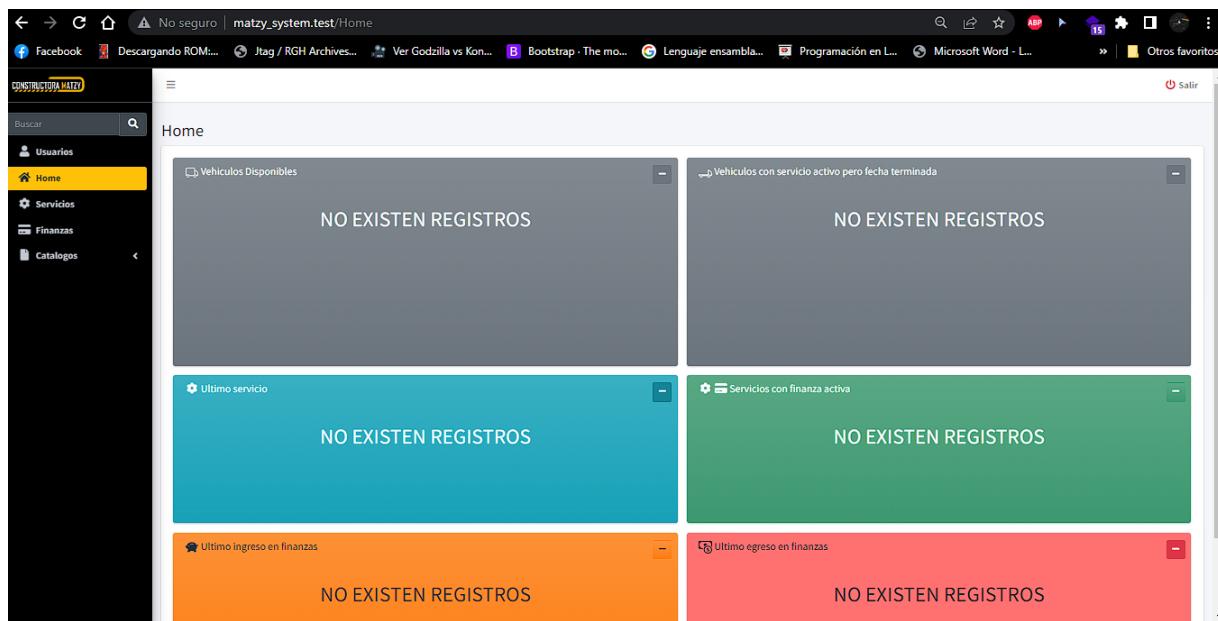


Figura 4.27 Interfaz de la mista “Home”, se mostrarán resúmenes de los datos.

Así es como se visualiza el sistema hasta este punto, las secciones son lo mismo, lo que cambia son colores y los iconos, los datos agregados serán mostrados de forma rápida en estas secciones.

SP1 H1 LIBERACIÓN Y ENTREGA DE PERFILES.

Entrega de Sprint

Información de la empresa y proyecto:

Empresa/ Organización	Constructora Matzy S.A de C.V.
Proyecto	Matzy.System

información del sprint:

Iteración/Sprint	ID	Enunciado de la historia
SP1 Perfiles	H1	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" de seguridad, con la finalidad de que solo yo pueda acceder a la información en el software y "modulo" que me permita tener datos resumidos del sistema en general, con la finalidad de que pueda ver dicha información al iniciar el software.

Información de la entrega:

Numero de iteración/ Sprint	SP1 H1/Perfiles
Persona responsable del recibimiento	[REDACTED]
Cargo	Dueño de la empresa

información del equipo de trabajo:

Nombre	Rol
Raul Guerrero Ramírez	Scrum master
Antonio Juárez Andrade	Equipo de trabajo

Observaciones:

Bajo el principio de Lean, sujeto a la o las demandas del cliente o dueño de la empresa, el sprint fue entregado con cada uno de los requerimientos citados en el enunciado de la historia, dando por concluido al cien por ciento el total de las tareas impuestas para el sprint SP1 H1 Perfiles.

Firma de conformidad del encargado
de recibir el sprint



Representante legal.
"Constructora Matzy"

Figura 4.28 Hoja de liberación del sprint perfiles.

TABLERO DE TAREAS FINAL DEL SP1 H1.

Historia de usuario	Tareas	En proceso	Concluido
H1 Perfiles	SP1 H1 T1		T1
H1 Perfiles	SP1 H1 T2		T2
H1 Perfiles	SP1 H1 T3		T3
H1 Perfiles	SP1 H1 T4		T4
H1 Perfiles	SP1 H1 T5		T5
H1 Perfiles	SP1 H1 T6		T6
H1 Perfiles	SP1 H1 T7		T7
H1 Perfiles	SP1 H1 T8		T8
Porcentaje de avance			
0-20%	21-40%	41-60%	61-80% 100%
			T2,T2,T3,T4,T5, T6,T7,T8

SPRINT BACKLOG SP2 H2.

ID	Product backlog
H2	Yo, "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita darles acceso a nuevos usuarios, con la finalidad de que mi personal administrativo pueda acceder a la información y/o crear nueva, una vez registrados.
ID	Iteración
H2	Sprint 2

ID	Tarea	Voluntario	Total, de Hrs.
T1	UX reglas para el funcionamiento del módulo de personal	Raul Guerrero Ramírez	32Hrs.
T2	UI maquetación del módulo principal	Raul Guerrero Ramírez	32Hrs.
T3	UI Maquetación de los modales para formulario	Raul Guerrero Ramírez	32Hrs.
T4	UI prototipado de maquetación para desarrollo	Raul Guerrero Ramírez	32Hrs.
T5	Programación de front-end y back-end del módulo “Usuarios”	Antonio Juarez Andrade	32Hrs.

Horas repartidas entre los días de desarrollo del sprint del 1 al 20																				
ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T1	8	8	8	8																
T2					8	8	8	8												
T3								8	8	8	8									
T4												8	8	8	8					
T5																8	8	8	8	

TABLERO DE TAREAS INICIO DEL SP2 H2.

Historia de usuario	Tareas	En proceso	Concluido
H2 Personal	SP2 H2 T1	T1	
H2 Personal	SP2 H2 T2	T2	
H2 Personal	SP2 H2 T3	T3	
H2 Personal	SP2 H2 T4	T4	
H2 Personal	SP2 H2 T5	T5	
Porcentaje de avance			
0-20%	21-40%	41-60%	61-80%
T1,T2,T3,T4,T5			
100%			

PRINCIPIO DE LEAN SP2 H2

El desarrollo de este sprint esta estrictamente sujeto a lo requerido en la historia de usuario, lo cual es crear un módulo dentro del sistema que permita agregar más usuarios al sistema y así, enfocando el desarrollo en la demanda y no en la oferta. Mediante este principio se evita la creación de funciones innecesaria que el usuario nunca usara y se agiliza los tiempos de desarrollo.

SP2 H2 T5 PROGRAMACIÓN DE FRONT-END Y BACK-END DE PERSONAL.

Para el módulo “Usuarios” el controlador destinado es “UsersController”, se agregaron métodos que son para mostrar, agregar, editar y eliminar los registros de los usuarios existentes del sistema, cada uno va acompañado de su front-end, la parte donde el usuario final interactúa con el sistema.

```

1  |?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use DB;
7
8  class UsersController extends Controller
9  {
10     public function __construct(){
11         $this->middleware('auth');
12     }
13
14
15 }
```

Figura 4.29 Código del controlador de “UsersController”.

El constructor ayuda a validar si el usuario ha iniciado sesión, esto con el fin de tener más seguridad en la integridad de los datos de la empresa verificando que el usuario que ingresa o modifica los datos es parte de la empresa.

```

16    public function view_users(){
17        $users=DB::table("users")->select("*")->get();
18        //return view('Users.users',compact("users"));
19        return view('users.users',compact("users"));
20    }
21 }
```

Figura 4.30 Código de la función “view_users”, retorna la vista “Usuarios”.

Este método retorna la vista del módulo “Usuarios”, se hace una consulta a la base de datos en específico a la tabla se “users” donde están todos los usuarios existentes, se envían a la vista mediante el “compact”.

```

126 <table class="table" style="width: 100%;">
127     <thead class="thead-dark">
128         <tr>
129             <th style="text-align: center;">Nombre</th>
130             <th style="text-align: center;">Usuario</th>
131
132         </tr>
133     </thead>
134     <tbody>
135         @foreach($users as $user)
136             <tr class="marca" onclick="pasar_id('{{ $user->id }}')">
137                 <td style="text-align: center;">
138                     {{ $user->name }}
139                 </td>
140                 <td style="text-align: center;">
141                     {{ $user->email }}
142                 </td>
143             </tr>
144         @endforeach
145     </tbody>
146 </table>
147 
```

Figura 4.31 Código de la estructura HTML de la vista para el módulo “Usuarios”.

Esta es la vista que retorna el controlador, es una tabla donde se muestran más a detalle los datos de cada uno de los usuarios y así poder eliminarlos, editarlos o agregar uno nuevo, para llenar la tabla se usó un “foreach” que permite recorrer el arreglo donde se encuentran los datos de los usuarios que se envió mediante el método del controlador.



Figura 4.32 Vista final del módulo “Usuarios”.

Para el filtro de la tabla y el paginado se usó una API de JavaScript “Datatable”, solo es agregar algunos links de CSS y de JavaScript se puede descargar, pero para este caso solo se jalan de internet.

```

o
9  <!-- estos son para la tabla-->
10 <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.12.1/css/dataTables.bootstrap5.min.css">
11 <link rel="stylesheet" type="text/css"
12 | href="https://cdn.datatables.net/responsive/2.3.0/css/responsive.dataTables.min.css">
13 @stop
14

358 @section('js')
359 <!-- estos son para la tabla-->
360 <script type="text/javascript" src="https://code.jquery.com/jquery-3.5.1.js"></script>
361 <script type="text/javascript" src="https://cdn.datatables.net/1.12.1/js/jquery.dataTables.min.js"></script>
362 <script type="text/javascript" src="https://cdn.datatables.net/1.12.1/js/dataTables.bootstrap5.min.js"></script>
363 <script type="text/javascript" src="https://cdn.datatables.net/responsive/2.3.0/js/dataTables.responsive.min.js">
364 </script>
365 <!-- este es para el selected2-->
366 <script src="https://cdn.jsdelivr.net/npm/select2@4.1.0-rc.0/dist/js/select2.min.js"></script>
367
368
369 <script type="text/javascript">
370
371     $(document).ready(function() {
372         $('.table').DataTable({
373             "language": {
374                 "url": "//cdn.datatables.net/plug-ins/1.10.15/i18n/Spanish.json"
375             }
376         });
377     });
378

```

Figura 4.33 Código JavaScript de la vista “Usuarios”, contiene links de la API y se indica la tabla afectada.

Para agregar un nuevo usuario se creó un formulario simple mediante un modal que el framework de Bootstrap proporciona, en la documentación indica como se debe de agregar al proyecto, los modales de editar y agregar serán muy similares solo que se cambian algunas etiquetas para que el usuario final identifique que es lo que está haciendo, si agrega o edita.

```

167 <form action="{{ url('/guardar_usuario') }}" method="POST">
168   @csrf
169   <div class="modal-body">
170     <div class="row">
171       <div class="col-md-6" style="margin-bottom: 10px;">
172         <label>Nombre</label>
173         <input type="text" name="nombre" class="form-control" id="nombre" onkeyup="activar_envio();" required>
174       </div>
175       <div class="col-md-6" style="margin-bottom: 10px;">
176         <label>Correo</label>
177         <input type="text" name="correo" class="form-control" id="correo" onkeyup="activar_envio();" required>
178       </div>
179     </div>
180     <div class="row" >
181       <div class="col-md-6" style="margin-bottom: 10px;">
182         <label>Contrasena</label>
183         <div class="input-group mb-3">
184           <input type="password" name="contrasena" id="contrasena" class="form-control" onkeyup="activar_envio();" required>
185           <button class="btn btn-outline-secondary" type="button" id="ver1" onclick="ver_contrasena();">
186             <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-eye-slash-fill" viewBox="0 0 16 16"></svg>
187           </button>
188         </div>
189       </div>
190       <div class="col-md-6" style="margin-bottom: 10px;">
191         <label>Repita Contrasena</label>
192         <div class="input-group mb-3">
193           <input type="password" name="contrasena-repit" id="contrasena2" class="form-control" onkeyup="activar_envio();" required>
194           <button class="btn btn-outline-secondary" type="button" id="ver2" onclick="ver_contrasena2();">
195             <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-eye-slash-fill" viewBox="0 0 16 16"></svg>
196           </button>
197         </div>
198       </div>
199       <div class="col-md-12">
200         <label id="texto_error" style="color: red;"></label>
201       </div>
202     </div>
203     <div class="modal-footer">
204       <button type="button" class="btn btn-danger" data-dismiss="modal">Cancelar</button>
205       <button class="btn" disabled id="envio_button">Aceptar</button>
206     </div>
207   </div>
208 </form>
209
210
211
212
213
214
215

```

Figura 4.34 Estructura HTML del modal para agregar un nuevo usuario, esta estructura será replicada para editar.

Este es el formulario que se encuentra dentro del modal de agregar nuevo usuario, solo son campos que el usuario debe de llenar, los datos a llenar son el nombre, correo, contraseña y repetir contraseña. Los campos de contraseñas se puede mostrar el contenido presionando un botón, para visualizar el texto ingresado.

```

379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
    function ver_contraseña(){
        if(document.getElementById("contraseña").type=="password"){
            document.getElementById("contraseña").type="text";
            document.getElementById("ver1").innerHTML='<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-eye-fill" viewBox="0 0 16 16"> <path d="M10.5 8a2.5 2.5 0 1 1-5 0 2.5 2.5 0 0 1 5 0z"/> <path d="M0 8s3-5.5 8-5.5S16 8 16 8s-5.5-8 8 0 8zm8 3.5a3.5 3.5 0 1 0 0-7 3.5 3.5 0 0 0 0 7z"/> </svg>';
        }else{
            document.getElementById("contraseña").type="password";
            document.getElementById("ver1").innerHTML='<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-eye-slash-fill" viewBox="0 0 16 16"> <path d="M10.79 12.912-1.614-1.615a3.5 3.5 0 0 1-4.474-4.474l-2.06-2.06C.938 6.278 0 8 0 8s3.5 8 5.5a7.029 7.029 0 0 0 2.79-.588zM5.21 3.088A7.028 7.028 0 0 1 8 2.5c5 0 8 5.5 8 5.5s-.939 1.721-2.641 3.238l-2.062-2.062a3.5 3.5 0 0 4-4.474-4.474L5.21 3.089z"/> <path d="M5.525 7.646a2.5 2.5 0 0 0 2.829 2.8291-2.83-2.829zm4.95.708-2.829-2.83a2.5 2.5 0 0 1 2.829zm3.171 6-12-12 .708-12 12-.708z"/> </svg>';
        }
    }

    function ver_contraseña2(){
        if(document.getElementById("contraseña2").type=="password"){
            document.getElementById("contraseña2").type="text";
            document.getElementById("ver2").innerHTML='<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-eye-slash-fill" viewBox="0 0 16 16"> <path d="M10.79 12.912-1.614-1.615a3.5 3.5 0 0 1-4.474-4.474l-2.06-2.06C.938 6.278 0 8 0 8s3.5 8 5.5a7.029 7.029 0 0 0 2.79-.588zM5.21 3.088A7.028 7.028 0 0 1 8 2.5c5 0 8 5.5 8 5.5s-.939 1.721-2.641 3.238l-2.062-2.062a3.5 3.5 0 0 4-4.474-4.474L5.21 3.089z"/> <path d="M5.525 7.646a2.5 2.5 0 0 0 2.829 2.8291-2.83-2.829zm4.95.708-2.829-2.83a2.5 2.5 0 0 1 2.829zm3.171 6-12-12 .708-12 12-.708z"/> </svg>';
        }else{
            document.getElementById("contraseña2").type="password";
            document.getElementById("ver2").innerHTML='<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-eye-slash-fill" viewBox="0 0 16 16"> <path d="M10.79 12.912-1.614-1.615a3.5 3.5 0 0 1-4.474-4.474l-2.06-2.06C.938 6.278 0 8 0 8s3.5 8 5.5a7.029 7.029 0 0 0 2.79-.588zM5.21 3.088A7.028 7.028 0 0 1 8 2.5c5 0 8 5.5 8 5.5s-.939 1.721-2.641 3.238l-2.062-2.062a3.5 3.5 0 0 4-4.474-4.474L5.21 3.089z"/> <path d="M5.525 7.646a2.5 2.5 0 0 0 2.829 2.8291-2.83-2.829zm4.95.708-2.829-2.83a2.5 2.5 0 0 1 2.829zm3.171 6-12-12 .708-12 12-.708z"/> </svg>';
        }
    }
}

```

Figura 4.35 Funciones de JavaScript para visualizar la contraseña ingresada por el usuario.

Estas son las funciones de JavaScript que se activan cuando se presiona el botón ver contraseña de los campos contraseña, lo que hacen estas funciones es mostrar el texto del campo oculto, solo cambia el tipo de campo y así es posible ver la contraseña. Las dos funciones hacen la misma función, pero una es para el campo contraseña y la otra es para el campo repetir contraseña, lo único que cambia son las etiquetas afectadas por dichas funciones.



Figura 4.36 Vista del modal para agregar un nuevo usuario, visualización de la función de ver contraseña.

Esta es la vista del formulario que será enviada al controlador de “UsersController”, con el método “agregar_users” se podrá registrar dicho registro a la base de datos.

```
1 public function agregar_users(Request $request){  
2  
3     try {  
4         DB::table("users")->insert([  
5             "name"=> $request["nombre"],  
6             "email"=> $request["correo"],  
7             "password"=> bcrypt($request["contrasena"]),  
8         ]);  
9  
10        return redirect()->back()->with(['message' => "Se Guardo correctamente el Usuario", 'color' => 'success']);  
11    } catch (\Exception $e) {  
12        return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);  
13    }  
14}
```

Figura 4.37 Método de PHP situada en el controlador “`UsersController`”, agrega un nuevo usuario a la DB.

El “try,catch” se usa por si falla la conexión a la base de datos y el usuario este informado de dicho problema para que pueda pedir mantenimiento al sistema y proporcione con más claridad que problema existe en el sistema. Se agrega a la tabla “users” el nuevo registro del usuario con la contraseña encriptada para más seguridad en el acceso, regresa a la vista anterior con un mensaje de éxito o de error si así pasara.



Figura 4.38 Vista del módulo “Usuarios”, mensaje de éxito al guardar el usuario.

Este es mensaje que muestra el sistema después de agregar el registro y se visualiza el nuevo registro. Para editar se ocupa las mismas funciones de JavaScript para mostrar la contraseña, pero cambiando algunas etiquetas para que el usuario sepa que está editando un registro, para poder editar o eliminar existe un menú escondido que sale si elijes o se hace clic en un registro de la tabla, estos botones activan los modales de eliminar o editar los registros.

```

221 <!--menu de opciones de la tabla-->
222 <div id="menu_opciones" class="visible_off" style="padding: 20px; background-color: #858585bd;">
223
224     <button type="button" class="close" style="margin-right: -17px; margin-top: -20px;" onclick="cerrar_menu();">
225         <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor" class="bi bi-x-octagon" viewBox="0 0 16 16" >=+
226             </svg>
227     </button>
228
229     <button class="btn btn-danger" style="margin-bottom: 10px; font-weight: bold; margin-top: 10px; width: 100%;" data-toggle="modal" data-target="#eliminar_usuario" onclick="eliminar_user();">
230         <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor" class="bi bi-pencil-square" viewBox="0 0 16 16" >=+
231             </svg>
232             Eliminar
233     </button>
234     <br>
235     <button class="btn btn-warning" style="margin-bottom: 10px; font-weight: bold; width: 100%;" data-toggle="modal" data-target="#editar_usuario" onclick="editar_user();">
236         <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor" class="bi bi-pencil-square" viewBox="0 0 16 16" >=+
237             </svg>
238             Editar
239     </button>
240     <br>
241 </div>
242
243
244
245
246
247
248

```

Figura 4.39 Estructura HTML del menú despegable de la tabla.

Este es el menú despegable de la tabla, solo contiene tres botones uno de ellos sirve para cerrar el menú y los otros son de editar y eliminar.

```

496 var id_user=null;
497
498 function pasar_id($id_tr) {
499     id_user=$id_tr;
500     var coordenadas_y=event.clientY; //obtenemos el valor de la posicion del boton
501     var coordenadas_x=event.clientX; //obtenemos el valor de la posicion del boton
502     menu_opciones.style.top=coordenadas_y-50+"px";
503     menu_opciones.style.left=coordenadas_x-50+"px";
504     menu_opciones.classList.add("visible_on");
505     menu_opciones.classList.remove("visible_off");
506     //alert($id_tr);
507 }
508
509 menu_opciones.addEventListener("mouseleave",function(){
510     menu_opciones.classList.remove("visible_on");
511     menu_opciones.classList.add("visible_off");
512 });
513
514 function cerrar_menu(){
515     menu_opciones.classList.remove("visible_on");
516     menu_opciones.classList.add("visible_off");
517 }
518

```

Figura 4.40 Función JavaScript que activa el menú despegable de la tabla.

Estas funciones son propias del menú, sirven para activar y cerrarlo, mediante la función de activar obtenemos el “id” del registro para poder saber a qué registro será editado o eliminado de la base de datos.

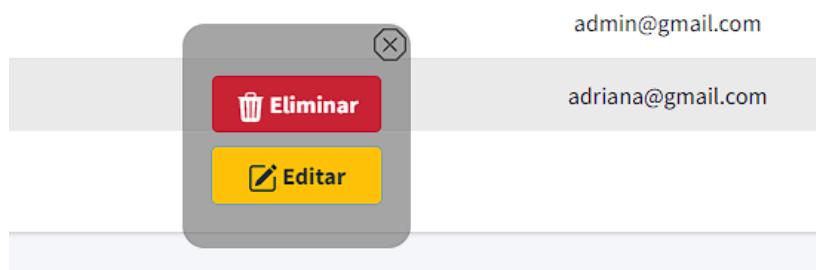


Figura 4.41 vista del menú despegable de la tabla con sus opciones.

```

519     function editar_user(){
520
521         $.ajax({
522             url: "{{url('/get_user')}}"+'/'+'id_user',
523             dataType: "json",
524             //context: document.body
525         }).done(function(user_data) {
526
527             if(user_data==null){
528                 document.getElementById("nombre_2").value=null;
529                 document.getElementById("correo_2").value=null;
530                 document.getElementById("id_user_edit").value=null;
531                 document.getElementById("envio_button_2").classList.remove("btn-warning");
532                 document.getElementById("envio_button_2").disabled = true;
533             }else{
534
535                 document.getElementById("nombre_2").value=user_data.name;
536                 document.getElementById("correo_2").value=user_data.email;
537                 document.getElementById("id_user_edit").value=user_data.id;
538                 document.getElementById("envio_button_2").classList.add("btn-warning");
539                 document.getElementById("envio_button_2").disabled = false;
540
541             }
542
543         });
544
545     }
546

```

Figura 4.42 Función de JavaScript que permite llenar los campos del modal con los datos del registro seleccionado.

Para el modal editar esta es una función que lo acompaña, dicha función es para llenar el modal de editar con los datos del registro seleccionado de la tabla, el llenado o la extracción de los datos de la base de datos se hace mediante Ajax, hace una consulta a una ruta creada del controlador “SearchesController”, este controlador hace búsquedas rápidas a la base de datos y retorna los datos en formato JSON.

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use DB;
7 class SearchesController extends Controller
8 {
9     public function __construct(){
10         $this->middleware('auth');
11     }
12
13     public function search_user($id){
14
15         $user_data=DB::table("users")->where("id",$id)->first();
16         return json_encode($user_data);
17     }
18
19

```

Figura 4.43 Método de PHP que está situada en el controlador “SearchesController”.

Esta es el método que retorna los datos de usuario, hace un filtro a la base de datos con el “id” mandado por la ruta marcada de Ajax.



Figura 4.44 vista del modal de editar usuario con los datos del registro seleccionado.

Así se muestra el modal de editar, es la misma plantilla usada para el registro, pero solo se cambiaron algunas etiquetas.

```

40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
public function actualizar_users(Request $request){
    try {
        if ($request["contrasena"] != null) {
            DB::table("users")->where("id", $request["id_user_edit"])->update([
                "name"=> $request["nombre"],
                "email"=> $request["correo"],
                "password"=> bcrypt($request["contrasena"]),
            ]);
        } else{
            DB::table("users")->where("id", $request["id_user_edit"])->update([
                "name"=> $request["nombre"],
                "email"=> $request["correo"],
            ]);
        }
        return redirect()->back()->with(['message' => "Se Actualizo correctamente el Usuario", 'color' => 'warning']);
    } catch (\Exception $e) {
        return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
    }
}

```

Figura 4.45 Método de PHP situada en el controlador “UsersController”, permite la actualización de un registro.

Este método es muy parecido que el de agregar, solo que se indica que será actualizado el registro con los datos enviados por el formulario del modal editar y de la misma forma regresa a la vista anterior con un mensaje de éxito o de error, según sea el caso.

Usuarios	
<p>Se Actualizo correctamente el Usuario</p>	
Agregar	
Mostrar 10 registros	Buscar: <input type="text"/>
Nombre	Usuario
ADMINISTRADOR	admin@gmail.com
ADRIANA JUANITA	adriana@gmail.com
Mostrando registros del 1 al 2 de un total de 2 registros	
Anterior 1 Siguiente	

Figura 4.46 Vista del mensaje de alerta de edición de un registro de la tabla.

El color del mensaje es amarillo, ya que es el color de precaución y el usuario reconoce.

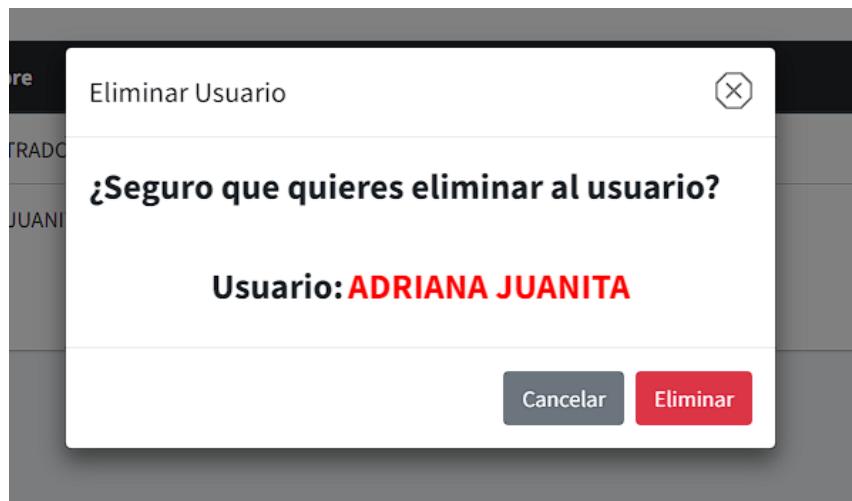


Figura 4.47 Modal de eliminar usuario de la DB del módulo “Usuarios”.

Este es el modal de eliminar, para eliminar un usuario de la base de datos solo es necesario el “id” del registro, se pone esta alerta para que el usuario este de acuerdo con el registro a eliminar de la base de datos definitivamente.

```

69
70     public function eliminar_user(Request $request){
71
72         try {
73
74             if ($request["id_user_edit"]!=0) {
75                 DB::table("users")->where("id",$request["id_user_edit"])->delete();
76
77             return redirect()->back()->with(['message' => "Se Elimino correctamente el Usuario", 'color' => 'danger']);
78
79         } catch (\Exception $e) {
80             return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
81
82         }
83
84     }
85
86 }
```

Figura 4.48 Método de PHP situada en el controlador “UsersController”, permite la eliminación de un registro.

Este método es donde se elimina por completo el registro de la base de datos y regresa a la vista anterior con un mensaje de éxito o de error.

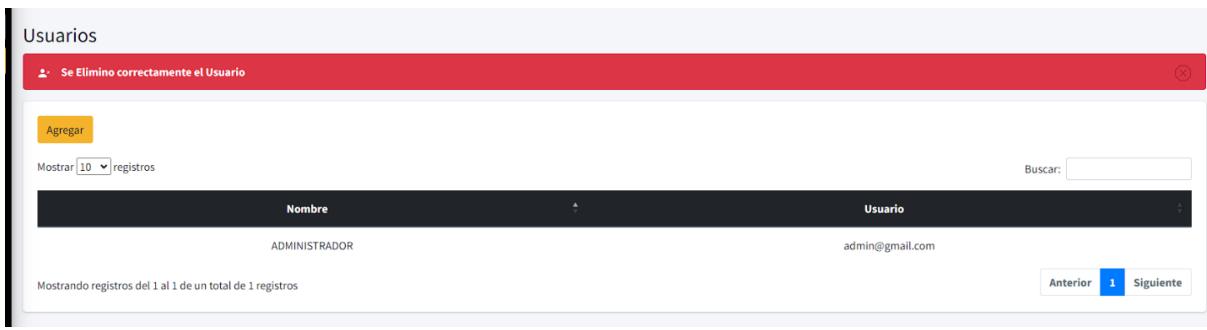


Figura 4.49 Vista del mensaje de alerta al eliminar un registro de la DB en el módulo “Usuarios”.

Las rutas del módulo de “Usuarios” son las siguientes.

```

41 //users
42 Route::get('/Users', [UserController::class,'view_users'])->name('users');
43 Route::post('/guardar_user',[UserController::class,'agregar_users'])->name('agregar_users');
44 Route::post('/Actualizar_user',[UserController::class,'actualizar_users'])->name('actualizar_users');
45 Route::delete('/Eliminar_user',[UserController::class,'eliminar_user'])->name('eliminar_user');
```

Figura 4.50 Rutas de activación de los métodos del controlador “UsersController”.

Cada ruta tiene la “url” por la cual el usuario final hará consulta y activa el método correspondiente de la clase “UsersController” que es el controlador de este módulo, este proceso será similar para todos los módulos ya que se trata de seguir el estilo de arquitectura de software (MVC).

SP2 H2 LIBERACIÓN Y ENTREGA DEL SPRINT PERSONAL.

Entrega de Sprint

Información de la empresa y proyecto:

Empresa/ Organización	Constructora Matzy S.A de C.V.
Proyecto	Matzy System

información del sprint:

Iteración/Sprint	ID	Enunciado de la historia
SP2 Personal	H2	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita darles acceso a nuevos usuarios, con la finalidad de que parte de mi personal pueda acceder a la información una vez yo le proporcione su propio usuario y contraseña.

Información de la entrega:

Numero de iteración/ Sprint	SP2 H2/Personal
Persona responsable del recibimiento	[REDACTED]
Cargo	Dueño de la empresa

información del equipo de trabajo:

Nombre	Rol
Raul Guerrero Ramírez	Scrum master
Antonio Juárez Andrade	Equipo de trabajo

Observaciones:

Bajo el principio de Lean, sujeto a la o las demandas del cliente o dueño de la empresa, el sprint fue entregado con cada uno de los requerimientos citados en el enunciado de la historia, dando por concluido al cien por ciento el total de las tareas impuestas para el sprint SP2 H2 Personal.

Firma de conformidad del encargado
de recibir el sprint



✓ Representante legal.
"Constructora Matzy"

Figura 4.51 Hoja de liberación del sprint “personal”.

TABLERO DE TAREAS FINAL DEL SP2 H2.

Historia de usuario	Tareas	En proceso	Concluido
H2 Personal	SP2 H2 T1		T1
H2 Personal	SP2 H2 T2		T2
H2 Personal	SP2 H2 T3		T3
H2 Personal	SP2 H2 T4		T4
H2 Personal	SP2 H2 T5		T5
Porcentaje de avance			
0-20%	21-40%	41-60%	61-80%
			T1,T2,T3, T4,T5
			100%

SPRINT BACKLOG SP3 H3.

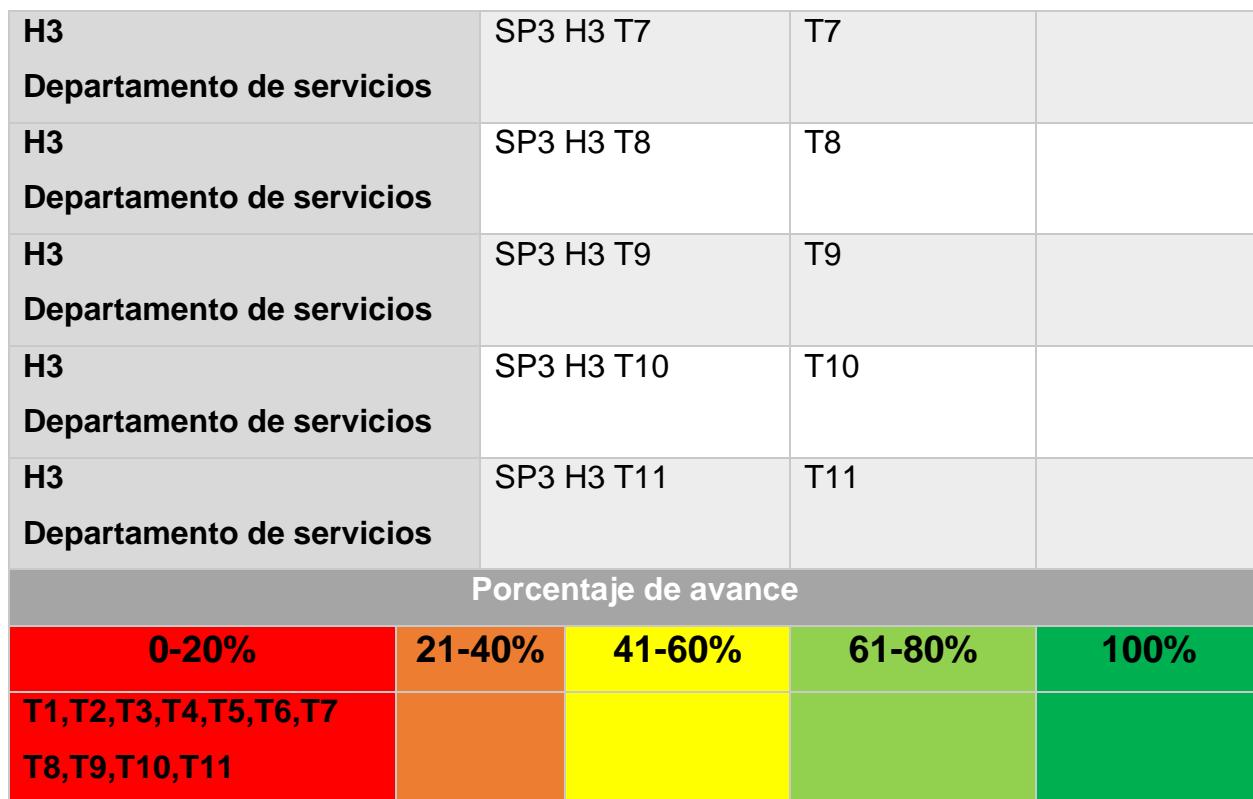
ID	Product backlog
H3	Yo, "representante legal de Constructora Matzy", necesito un "módulo" que me permita recabar los datos necesarios del cliente y verificar la disponibilidad de mis vehículos y las fechas en que estarán disponibles, y de esta manera, puedo decidir correctamente cuándo iniciar un servicio, obteniendo un informe interno con todo lo registrado y a la vez uno que me permita registrar mis vehículos, con el fin de registrar el vehículo una sola vez y no cada nuevo servicio.
ID	Iteración
H3	Sprint 3

ID	Tarea	Voluntario	Total, de Hrs.
T1	Definir las reglas de negocio para el correcto funcionamiento.	Raul Guerrero Ramírez	12Hrs.
T2	Definir datos que implican una constante en la empresa.	Raul Guerrero Ramírez	12Hrs.
T3	Diseño para hoja o membrete para reportes.	Raul Guerrero Ramírez	12Hrs.
T4	Maquetación de apartado “catálogos”	Raul Guerrero Ramírez	12Hrs.
T5	Maquetación del módulo de servicios.	Raul Guerrero Ramírez	12Hrs.
T6	Prototipado de catálogos.	Raul Guerrero Ramírez	12Hrs.
T7	Prototipado de módulo de servicios.	Raul Guerrero Ramírez	12Hrs.
T8	Envío de correos por parte del sistema.	Antonio Juarez Andrade	12Hrs.
T9	Programación de front-end y back-end del módulo “Catálogos”	Antonio Juarez Andrade	12Hrs.
T10	Programación de front-end y back-end del módulo “Servicios”	Antonio Juarez Andrade	12Hrs.
T11	Reportes.	Antonio Juarez Andrade	12Hrs.

		Horas repartidas entre los días de desarrollo del sprint del 1 al 20																				
ID		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
T1		3	3	3	3																	
T2						3	3	3	3													
T3									3	3	3	3										
T4															3	3	3	3				
T5																		3	3	3	3	
T6																			3	3	3	3
T7										3	3	3	3									
T8		3	3	3	3																	
T9						3	3	3	3													
T10									3	3	3	3										
T11															3	3	3	3				

TABLERO DE TAREAS INICIO DEL SP3 H3.

Historia de usuario	Tareas	En proceso	Concluido
H3 Departamento de servicios	SP3 H3 T1	T1	
H3 Departamento de servicios	SP3 H3 T2	T2	
H3 Departamento de servicios	SP3 H3 T3	T3	
H3 Departamento de servicios	SP3 H3 T4	T4	
H3 Departamento de servicios	SP3 H3 T5	T5	
H3 Departamento de servicios	SP3 H3 T6	T6	



PRINCIPIO DE LEAN SP3 H3

El desarrollo de este sprint está estrictamente sujeto a lo requerido en la historia de usuario que es crear un módulo donde se agreguen vehículos y así poder crear nuevos servicios a partir de los vehículos agregados, verificando su existencia y disponibilidad, así como generar un PDF del servicio. Mediante este principio se evita la creación de funciones innecesarias que el usuario nunca utilizará y se agilizan los tiempos de desarrollo.

SP3 H3 T8 ENVIO DE CORREOS.

El sistema enviara correos mediante una función de Laravel, este enviara información sobre movimientos en el sistema, la clase que se encargara del envío de correos se crea mediante un comando en el “Cmder” o crearla directamente. Los correos serán un poco personalizados, tendrá imágenes referentes al sistema y títulos referentes a los módulos.

Esta clase es capaz de enviar una vista de HTML, para ellos se crearán tres vistas de HTML para cada uno de los módulos principales del sistema.

```

3  namespace App\Mail;
4
5  use Illuminate\Bus\Queueable;
6  use Illuminate\Contracts\Queue\ShouldQueue;
7  use Illuminate\Mail\Mailable;
8  use Illuminate\Queue\SerializesModels;
9
10 class MessageSent extends Mailable
11 {
12     use Queueable, SerializesModels;
13
14     public $datos;
15     public $subject;
16     public $modulo;
17     public function __construct($titulo,$data,$modulo)
18     {
19         $this->datos = $data;
20         $this->subject = $titulo;
21         $this->modulo = $modulo;
22     }
23
24     public function build()
25     {
26         if($this->modulo=="service"){
27             return $this->view('Emails.service_new');
28         }
29
30         if($this->modulo=="finance"){
31             return $this->view('Emails.finance_new');
32         }
33
34         if($this->modulo=="catalogos"){
35             return $this->view('Emails.catalogos_new');
36         }
37     }

```

Figura 4.52 Clase de PHP que permite el envío de correos electrónicos, implementación de Laravel.

Esta clase pide como parámetros tres campos uno que es el título otro que es la información que tendrá el correo y uno que indica a qué modulo pertenece esos datos y que vista usará la clase. Las vistas son para los módulos de servicios, finanzas y catálogos dependiendo el tipo es la vista que enviará al correo, la construcción de cada vista es muy simple, y similar entre ellas, solo cabían títulos y el acomodo de los textos.

```

7   </head>
8   <body style="background-color: #E8E7E2; padding-top: 10px; padding-bottom: 10px;">
9
10  <div style="margin-right: 25%; margin-left: 25%; margin-top: 20px; margin-bottom: 20px; border-radius: 10px; background-color: white;">
11
12
13      
14
15      <div style="margin-bottom: 20px; text-align: center;">
16          <label style="font-weight: 35px; width: 100%; font-family: 'Avant Garde', Avantgarde, 'Century Gothic', CenturyGothic, 'AppleGothic', sans-serif; font-size: 30px;">
17              @if($datos['tipo'] == "nuevo")
18                  Se creo un registro nuevo en catalogos
19              @endif
20              @if($datos['tipo'] == "actualizar")
21                  Se actualizo un registro de catalogos
22              @endif
23              @if($datos['tipo'] == "eliminar")
24                  Se elimino un registro de catalogos
25              @endif
26      </label>
27  </div>
28  <br><br>
29  <div style="margin-bottom: 20px; text-align: center; font-weight: 35px; width: 100%; font-family: 'Avant Garde', Avantgarde, 'Century Gothic', CenturyGothic, 'AppleGothic', sans-serif; font-size: 20px;">
30      <label>Hecho por: {{$datos['usuario']}}</label><br>
31      <label>Matricula: {{$datos['matricula']}}</label><br>
32      <label>Nombre: {{$datos['nombre']}}</label><br>
33      <label>Funcion: {{$datos['funcion']}}</label><br>
34      @if($datos['tipo'] != "nuevo" && $datos['tipo'] != "eliminar")
35          <label>**PUEDES CREAR UN SERVICIO SIN PROBLEMAS</label><br>
36      @endif
37  </div>
38  <br>
39  <br>
40  <div style="margin-bottom: 20px; text-align: center;">
41      <a href="{{url('/catalogos')}}" style="border-radius: 10px; text-decoration: none; font-size: 35px; background-color: #C6B830; color: black; width: 100%; padding: 15px;">
42          VER REGISTROS
43          
44      </a>
45  </div>
46  <br><br>
47
48
49  </div>
50
51 </body>
-->
```

Figura 4.53 Estructura HTML de la vista que tendrá el correo electrónico.

Esta es una de las vistas que envía el correo, en específico es la vista que enviará si es el módulo de “Catálogos”, que es el módulo que se construirá después, la arquitectura de esta vista es la misma para las demás, pero solo cambia el acomodo del texto y los datos a mostrar, una vez hecho esto ya podemos enviar correos mediante Laravel. Los correos serán enviados mediante un correo de Google, para poder enviar correos mediante este servicio se debe de crear una clave de acceso de aplicación en la cuenta que se va a usar, en este caso se creó un correo especial para el sistema y se generó la clave, también se puede con otro tipo de servicio como el de Hotmail, pero tienen límite de envíos por día.

SP3 H3 T9 PROGRAMACIÓN DE FRONT-END Y BACK-END DE CATÁLOGOS.

El controlador destinado para El módulo de “catálogos” es “CatalogosController”. Este módulo es muy importante para la funcionalidad del sistema, este módulo se encarga de agregar los vehículos que posteriormente se hace uso en el módulo de “servicios”, tener un apartado especial para agregar estos vehículos, hace que el sistema sea más limpio y ordenado, se evita la duplicidad de dato o en este caso de vehículos, este módulo servirá para agregar más cosas de “almacén” que la empresa en un futuro tenga que agregar, por ejemplo, piezas de refacción para los vehículos.

Este es un módulo muy simple ya que es muy similar que el módulo de “Personal”, se agregaron métodos que son para mostrar, agregar, editar y eliminar los registros de los vehículos existentes del sistema, cada uno va acompañado de su front-end, la parte donde el usuario final interactúa con el sistema.

```
22     public function view_catalogos(){
23         $vehiculos=DB::table("vehiculos")->select("*")->get();
24
25         return view('Catalogos.vehiculos',compact('vehiculos'));
26     }
27 }
```

Figura 4.54 Función PHP situada en el controlador “CatalogosController”, retorna la vista del módulo “Catálogos”.

Este método retorna la vista del módulo “Catálogos”, se hace una consulta a la base de datos en específico a la tabla se “vehículos” donde están todos los vehículos existentes y donde serán registrados todos, se envían a la vista mediante el “compact”.

```

113
114 <div class="card">
115   <div class="card-body">
116     <div style="margin-bottom: 20px;">
117       <button class="btn" style="background-color: #F5B32B" data-toggle="modal" data-target="#agregar" >Agregar</button>
118     </div>
119     <div class="table-responsive">
120
121       <table class="table" style="width: 100%;">
122         <thead class="thead-dark">
123           <tr>
124             <th style="text-align: center;">Matricula</th>
125             <th style="text-align: center;">Nombre</th>
126             <th style="text-align: center;">Funcion</th>
127
128           </tr>
129         </thead>
130         <tbody>
131           @foreach($vehiculos as $vehiculo)
132           <tr class="marca" onclick="pasar_id('{{ $vehiculo->id }})">
133             <td style="text-align: center;">
134               {{$vehiculo->matricula}}
135             </td>
136             <td style="text-align: center;">
137               {{$vehiculo->nombre}}
138             </td>
139             <td style="text-align: center;">
140               {{$vehiculo->funcion}}
141             </td>
142
143           </tr>
144           @endforeach
145         </tbody>
146       </table>
147
148     </div>
149   </div>
150 </div>
151
152

```

Figura 4.55 Estructura HTML de la vista “Catálogos”.

Esta es la vista que retorna el controlador, es una tabla donde se muestran más a detalle los datos de cada uno de los vehículos y así poder eliminarlos, editarlos o agregar uno nuevo, para llenar la tabla se usó un “foreach” que permite recorrer el arreglo donde se encuentran los datos de los vehículos que se envió mediante el método del controlador, como se pude apreciar solo son tres datos lo que se muestran y que en base datos solo se le agrega el “id”.

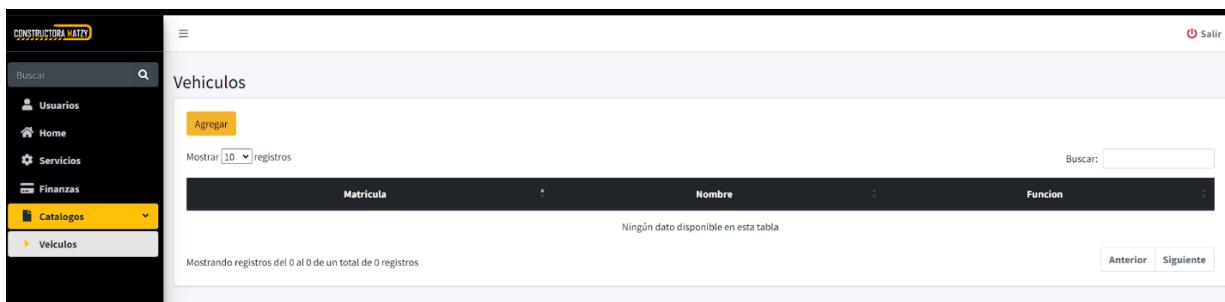


Figura 4.56 Vista final del módulo “Catálogos”.

Para el filtro de la tabla y el paginado se usó nuevamente la API de JavaScript “Datatable” que se agregó en el módulo de “Usuarios”.

Para agregar un nuevo vehículo se creó un formulario simple mediante un modal de la misma manera que se usó para el módulo anterior, los modales de editar y agregar serán muy similares solo que se cambian algunas etiquetas para que el usuario final identifique que es lo que está haciendo si agrega, eliminar o edita.

```

154  <!-- modal de agregar vehiculo-->
155  <div class="modal fade" id="agregar" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
156      <div class="modal-dialog modal-dialog-centered">
157          <div class="modal-content">
158              <div class="modal-header">
159                  <h5 class="modal-title" id="exampleModalLabel">Agregar Vehiculo</h5>
160                  <button type="button" class="close" data-dismiss="modal" aria-label="Close">...
161              </div>
162              <form action="{{ url('/guardar_vehiculo') }}" method="POST">
163                  @csrf
164                  <div class="modal-body">
165                      <div class="row">
166                          <div class="col-md-6" style="margin-bottom: 10px;">
167                              <label>Matricula</label>
168                              <input type="text" name="matricula" class="form-control" id="matricula" onkeyup="activar_envio(); required>
169                          </div>
170                          <div class="col-md-6" style="margin-bottom: 10px;">
171                              <label>Nombre</label>
172                              <input type="text" name="nombre" class="form-control" id="nombre" onkeyup="activar_envio(); required>
173                          </div>
174                      </div>
175                      <div class="col-md-12">
176                          <label>Funcion</label>
177                          <textarea name="funcion" class="form-control" id="funcion" onkeyup="activar_envio(); required></textarea>
178                      </div>
179                  </div>
180                  <div class="modal-footer">
181                      <button type="button" class="btn btn-danger" data-dismiss="modal">Cancelar</button>
182                      <button class="btn" disabled id="envio_button">Aceptar</button>
183                  </div>
184              </form>
185          </div>
186      </div>
187  </div>
188
189
190
191
192
193
194

```

Figura 4.57 Estructura HTML del modal para agregar un vehículo a la DB.

Este es el formulario que se encuentra dentro del modal de agregar nuevo vehículo que se activa mediante el botón de agregar, solo son campos que el usuario debe de llenar, los datos a llenar son la matrícula, nombre y función.

```

328  function activar_envio(){
329
330      if(document.getElementById("matricula").value!="" && document.getElementById("nombre").value!="" && document.getElementById("funcion").value!
331      != ""){
332
333          document.getElementById("envio_button").classList.add("btn-success");
334          document.getElementById("envio_button").disabled = false;
335
336      }else{
337          document.getElementById("envio_button").classList.remove("btn-success");
338          document.getElementById("envio_button").disabled = true;
339      }
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
789

```

Figura 4.58 Función de JavaScript que valida si los campos del modal no están vacíos.

Esta es una función del formulario anterior, lo único que hace es verificar si los campos que se le piden al usuario no estén vacíos, esto con el fin de que el usuario no envíe datos vacíos y así evitar errores con la base de datos.

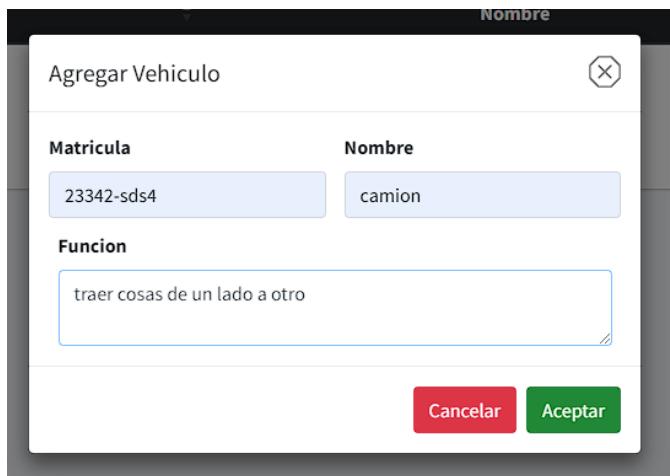


Figura 4.59 Vista del modal de agregar un nuevo vehículo a la DB.

Esta es la vista del formulario que será enviada al controlador de “CatalogosController”, con el método “agregar_vehiculo” se podrá registrar dicho registro a la base de datos.

```

29 public function agregar_vehiculo(Request $request){
30
31
32
33
34     try {
35
36         DB::table("vehiculos")->insert([
37             "matricula"=> $request["matricula"],
38             "nombre"=> $request["nombre"],
39             "funcion"=>$request["funcion"],
40         ]);
41
42
43         $data =["usuario" => Auth::user()->name,"matricula" => $request["matricula"],"nombre" => $request["nombre"],"tipo" => "nuevo","funcion"
44             => $request["funcion"]];
45         Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Nuevo Registro en Catalogos",$data,"catalogos"));
46
47         return redirect()->back()->with(['message' => "Se Guardo correctamente el Vehiculo", 'color' => 'success']);
48
49     } catch (\Exception $e) {
50         return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
51     }
52 }
```

Figura 4.60 Método de PHP situada en el controlador “CatalogosController”, permite agregar un nuevo registro.

El “try,catch” se usa por si falla la conexión a la base de datos y el usuario este informado de dicho problema para que pueda pedir mantenimiento al sistema y proporcione con más claridad que problema existe en el sistema. Se agrega a la tabla

“Vehículos” el nuevo registro del vehículo del formulario enviado, regresa a la vista anterior con un mensaje de éxito o de error si así pasara.

```
$data =["usuario" => Auth::user()->name,"matricula" => $request["matricula"],"nombre" => $request["nombre"],"tipo" => "nuevo","funcion" => $request["funcion"]];
Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Nuevo Registro en Catalogos",$data,"catalogos"));
```

Figura 4.61 Código PHP que activa el envío de correo electrónico.

Estas líneas que se agregaron en el método son para el envío de correos por parte del sistema, de esta manera le pasamos los datos a la clase que envía los correos e indicamos a que correo serán enviados estos datos, como todavía está en desarrollo se envían a un correo de pruebas, después se agregara el correo de la empresa o de alguna persona a cargo.



Figura 4.62 Vista final del correo cuando se agrega un nuevo vehículo a la DB.

Este es el correo que se envía, se agregó un link para que el usuario pueda ir directamente a los registros agregados en ese apartado o modulo, este formato será similar para los demás movimientos dentro de este módulo.

The screenshot shows a web-based application interface for managing vehicles. At the top, there's a green header bar with the text "Se Guardo correctamente el Vehículo". Below this, there's a table with columns: Matrícula, Nombre, and Función. A single row is visible with the values "23342-sds4", "camion", and "traer cosas de un lado a otro". At the bottom of the table, it says "Mostrando registros del 1 al 1 de un total de 1 registros". On the left side of the table, there's a yellow "Agregar" button. On the right side, there are navigation buttons for "Anterior" and "Siguiente".

Figura 4.63 Vista del mensaje de éxito al agregar un nuevo vehículo a la DB.

Este es mensaje que muestra el sistema después de agregar el registro y se visualiza el nuevo registro.

Para editar es el mismo modal que el de agregar, pero se cambiaron algunas etiquetas, para activar el modal de editar, es mediante el menú despegable de la tabla, es el mismo menú usado para el módulo “Usuarios”, se ocupa el mismo para que el usuario sepa que cuando vea una tabla sepa que tiene un menú despegable y para el programador es más fácil porque solo es reciclar código.

The screenshot shows a modal dialog titled "Editar Vehiculo". It contains three input fields: "Matrícula" with the value "23342-sds4", "Nombre" with the value "camion", and "Función" with the value "nada". At the bottom of the modal are two buttons: "Cancelar" (in red) and "Actualizar" (in yellow).

Figura 4.64 vista del modal editar un vehículo situado en el módulo “Catálogos”.

Así se muestra el modal de editar, es la misma plantilla usada para el registro, pero solo se cambiaron algunas etiquetas, para llenar los campos con sus respectivos datos de igual manera se usó Ajax para consultar el registro y llenar los datos.

```

376 function editar_registro(){
377
378     $.ajax({
379         url: "{url('/get_vehiculo')}"+'/' + id_vehiculo,
380         dataType: "json",
381         //context: document.body
382     }).done(function(vehiculo_data) {
383
384         if(vehiculo_data==null){
385             document.getElementById("funcion2").value=null;
386             document.getElementById("nombre2").value=null;
387             document.getElementById("matricula2").value=null;
388             document.getElementById("id_vehiculo_edit").value=null;
389             document.getElementById("envio_button_2").classList.remove("btn-warning");
390             document.getElementById("envio_button_2").disabled = true;
391         }else{
392
393             document.getElementById("funcion2").value=vehiculo_data.funcion;
394             document.getElementById("nombre2").value=vehiculo_data.nombre;
395             document.getElementById("matricula2").value=vehiculo_data.matricula;
396             document.getElementById("id_vehiculo_edit").value=vehiculo_data.id;
397             document.getElementById("envio_button_2").classList.add("btn-warning");
398             document.getElementById("envio_button_2").disabled = false;
399
400

```

Figura 4.65 Función de JavaScript que llena los campos con los datos del registro seleccionado.

Esta la función que se activa al abrir el modal de editar, lo que hace es llenar los campos con los datos del registro seleccionado en la tabla mediante el menú desplegable.

```

52
53     public function actualizar_vehiculo(Request $request){
54
55         try {
56
56             DB::table("vehiculos")->where("id",$request["id_vehiculo_edit"])->update([
57                 "matricula"=> $request["matricula"],
58                 "nombre"=> $request["nombre"],
59                 "funcion"=>$request["funcion"],
60             ]);
61
62
63             $data =[ "usuario" => Auth::user()->name,"matricula" => $request["matricula"], "nombre" => $request["nombre"], "tipo" => "actualizar",
64             "funcion" => $request["funcion"]];
65             Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Registro Actualizado en Catalogos",$data,"catalogos"));
66
67
68             return redirect()->back()->with(['message' => "Se Actualizo correctamente el Vehiculo", 'color' => 'warning']);
69
70         } catch (\Exception $e) {
71             return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
72         }
73
74     }
75

```

Figura 4.66 Método de PHP situada en el controlador “CatalogosController”, permite actualizar un registro.

Este método es muy parecido que el de agregar, pero solo se indica que será actualizado el registro con los datos enviados por el formulario del modal editar, una vez que termina se envía el correo como el enviado anteriormente al agregar el registro, pero solo se indica que se actualizo y regresamos a la vista anterior con un mensaje de éxito o de error, según sea el caso.

The screenshot shows a table with three columns: Matricula, Nombre, and Funcion. One row is visible with the values '23342-sds4', 'camion', and 'nada'. At the top, a yellow banner displays the message 'Se Actualizo correctamente el Vehiculo'. Below the table, there are buttons for 'Anterior' and 'Siguiente'.

Figura 4.67 Vista del mensaje de alerta al editar un registro en el módulo “Catálogos”.

El sistema en todo momento muestra mensajes de alerta y con colores como en este caso de color amarillo para que el usuario los identifique.

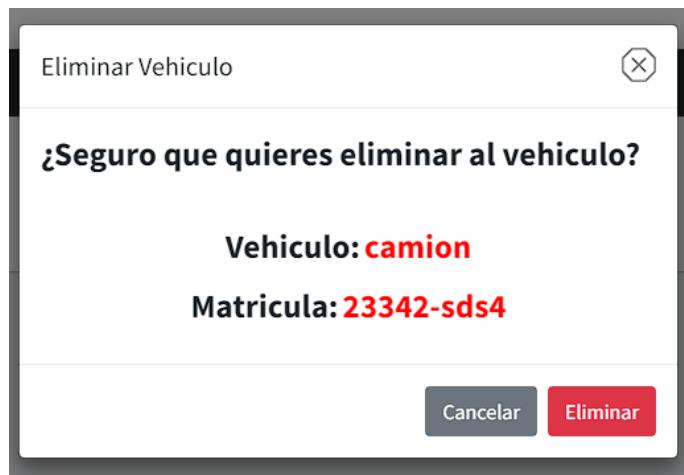


Figura 4.68 Vista del modal de eliminar un vehículo de la DB.

Este es el modal de eliminar, para eliminar un vehículo de la base de datos solo es necesario el “id” del registro, se pone esta alerta para que el usuario este de acuerdo con el registro a eliminar de la base de datos definitivamente.

```

75
76     public function eliminar_vehiculo(Request $request){
77
78         try {
79
80             if ($request["id_vehiculo_edit"]!=0) {
81
82                 $vehiculo=DB::table("vehiculos")->where("id",$request["id_vehiculo_edit"])->first();
83                 DB::table("vehiculos")->where("id",$request["id_vehiculo_edit"])->delete();
84
85                 $data =["usuario" => Auth::user()->name,"matricula" => $vehiculo->matricula,"nombre" => $vehiculo->nombre,"tipo" => "eliminar",
86                         "funcion" => $vehiculo->funcion];
87                 Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Registro Actualizado en Catalogos",$data,"catalogos"));
88
89                 DB::table("vehiculos")->where("id",$request["id_vehiculo_edit"])->delete();
90
91             }
92
93             return redirect()->back()->with(['message' => "Se Elimino correctamente el Vehiculo", 'color' => 'danger']);
94
95         } catch (\Exception $e) {
96             return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
97         }
98
99     }
100 }

```

Figura 4.69 Método de PHP situado en el controlador “CatalogosController”, permite eliminar un registro.

Este es el método encargado de su eliminación del registro, también este método envía un correo de alerta para que el administrador sepa que algo fue eliminado, elimina el registro de la base de datos y regresa a la vista anterior con un mensaje de éxito o de error.



Figura 4.70 Vista del mensaje de alerta al eliminar un vehículo de la DB.

Las rutas del módulo de “Catálogos” son las siguientes.

```

48 //catalogos
49 Route::get('/Catalogos',[CatalogosController::class,'view_catalogos'])->name('catalogos');
50 Route::post('/guardar_vehiculo',[CatalogosController::class,'agregar_vehiculo'])->name('agregar_vehiculo');
51 Route::post('/Actualizar_vehiculo',[CatalogosController::class,'actualizar_vehiculo'])->name('actualizar_vehiculo');
52 Route::delete('/Eliminar_vehiculo',[CatalogosController::class,'eliminar_vehiculo'])->name('eliminar_vehiculo');
53

```

Figura 4.71 Rutas que activan los métodos del Controlador “CatalogosController”.

Cada ruta tiene la “url” por la cual el usuario final hará consulta y activa el método correspondiente de la clase “CatalogosController” que es el controlador de este módulo.

SP3 H3 T10 PROGRAMACIÓN DE FRONT-END Y BACK-END DE SERVICIOS.

Para el correcto funcionamiento del módulo “servicios” debe de estar listo el módulo “Catálogos” ya que el módulo de “Servicios” depende de los vehículos agregados, como el módulo “Catálogos” ya está terminado solo se agregarán algunos registros como ejemplo para poder trabajar con el módulo “Servicios”.

El controlador destinado para este módulo es “ServicesController” y de igual manera se agregaron los métodos de agregar, editar y eliminar como en los otros módulos.

```

~^
22 public function view_services(){
23
24     date_default_timezone_set('America/Mexico_City');
25     $services=DB::table("servicios")->select("*")->get();
26     $ultimo_id=DB::table("servicios")->latest('id')->first();
27     $servicio_vehiculos=DB::table("servicio_vehiculo")->select("*")->get();
28     $i=0;
29     $filtro_ocupadas=null;
30     foreach ($servicio_vehiculos as $contador) {
31         $filtro_ocupadas[$i]=$contador->id_vehiculo;
32         $i++;
33     }
34
35
36     $vehiculos=null;
37     if ($filtro_ocupadas==null) {
38         $vehiculos=DB::table("vehiculos")->select("*")->get();
39         $vehiculos_servicio_activo=null;
40     }else{
41         $vehiculos=DB::table("vehiculos")->whereNotIn("id",$filtro_ocupadas)->get();
42         $vehiculos_servicio_activo=DB::table("servicio_vehiculo_registro")->whereIn("id_vehiculo",$filtro_ocupadas)->whereDate("fecha_fin","<",
43             date("Y-m-d"))->get();
44     }
45
46     return view('Services.services',compact("vehiculos","ultimo_id","services","vehiculos_servicio_activo","filtro_ocupadas","servicio_vehiculos"));
47 }
```

Figura 4.72 Método de PHP situado en el controlador “ServicesController”, retorna la vista del módulo “Servicios”.

Ese método es el que retorna la vista del módulo “Servicios” , se hace una consulta a la base de datos en específico a la tabla se “servicios” donde están todos los servicios existentes y donde serán registrados todos, se envían a la vista mediante el “compact” y también se agregaron otras consultas como la consulta a la tabla “servicio_vehiculo” que es la que indica los vehículos seleccionados para los servicios.

```

124 <div class="card">
125   <div class="card-body">
126     <div style="margin-bottom: 20px;">
127       <button class="btn" style="background-color: #F5B32B" data-toggle="modal" data-target="#agregar" >Agregar</button>
128     </div>
129     <div class="table-responsive">
130
131       <table class="table" style="width: 100%;">
132         <thead class="thead-dark">
133           <tr>
134             <th style="text-align: center;">Folio</th>
135             <th style="text-align: center;">No. Vehículos</th>
136             <th style="text-align: center;">Cliente</th>
137             <th style="text-align: center;">F. Inicio</th>
138             <th style="text-align: center;">F. Termino</th>
139             <th style="text-align: center;">Estatus</th>
140
141           </tr>
142         </thead>
143         <tbody>
144           @foreach($services as $service)
145             <tr class="marca" onclick="pasar_id('{{ $service->id }}')>
146               <td style="text-align: center;">
147                 CMZ-{{ $service->id }}
148               </td>
149               <td style="text-align: center;">
150                 <?php $contador=0; ?>
151                 @foreach($servicio_vehiculos as $servicio_vehiculo)
152                   @if($servicio_vehiculo->id_servicio==$service->id)
153                     <?php $contador++; ?>
154                   @endif
155                   @endforeach
156                   {{ $contador }}
157               </td>
158               <td style="text-align: center;">
159                 {{ $service->nombre }}
160               </td>
161               <td style="text-align: center;"> ...
162               </td>
163               <td style="text-align: center;"> ...
164               </td>
165               <td style="text-align: center;">
166                 espera
167               </td>
168             </tr>
169           @endforeach
170         </tbody>
171     </table>
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201

```

Figura 4.73 Estructura HTML de la vista para el módulo “Servicios”.

Esta es la vista que retorna el controlador, es una tabla donde se muestran más a detalle los datos de cada uno de los servicios y así poder eliminarlos, editarlos o agregar uno nuevo, para llenar la tabla se usó un “foreach” que permite recorrer el arreglo donde se encuentran los datos de los servicios que se envió mediante el método del controlador.



Figura 4.74 Vista final del módulo “Servicios”.

Para el filtro de la tabla y el paginado se usó nuevamente la API de JavaScript “Datatable” que se agregó en el módulo de “Usuarios”.

Para agregar un nuevo servicio se creó un formulario simple mediante un modal de la misma manera que se usó para el módulo anterior, los modales de editar y agregar serán muy similares solo que se cambian algunas etiquetas para que el usuario final identifique que es lo que está haciendo si agrega, eliminar o edita.

```

208 <!-- modal de agregar servicio-->
209 <div class="modal fade" id="agregar" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
210   <div class="modal-dialog modal-xl modal-dialog-centered">
211     <div class="modal-content">
212       <div class="modal-header">
213         <h5 class="modal-title" id="exampleModalLabel">Agregar Servicio</h5>
214         <button type="button" class="close" data-dismiss="modal" aria-label="Close">...
215       </button>
216     </div>
217     <form action="{{ url('/guardar_servicio') }}" method="POST">
218       @csrf
219       <div class="modal-body">
220         <div class="row">
221           <div class="col-md-9" style="margin-bottom: 10px;">
222             <div class="row">
223               <div class="col-md-3" style="margin-bottom: 10px;">
224                 <label>Folio</label>
225                 <input style="font-weight: bold; font-size: 20px;" type="text" name="folio" class="form-control" id="folio" onkeyup="...
226                   activar_envio();" value="CMZ-<?php if($ultimo_id->null){echo 1;}else{echo $ultimo_id->id+1;} ?>" required disabled>
227               </div>
228             </div>
229           </div>
230         </div>
231         <div class="row"> ...
232         </div>
233         <div class="row"> ...
234         </div>
235         <br>
236         <div class="row"> ...
237         </div>
238
239         <div class="row"> ...
240         </div>
241         <div class="row"> ...
242         </div>
243
244         <div id="contenedor_padre">
245           <div class="col-md-12" style="margin-bottom: 10px;">
246             <label>Observaciones Generales</label>
247             <textarea name="observaciones_g" class="form-control" onkeyup="activar_envio();"></textarea>
248           </div>
249         </div>
250         <div class="modal-footer">
251           <input type="hidden" name="cantidad_vehiculos" id="cantidad_vehiculos">
252           <button type="button" class="btn btn-danger" data-dismiss="modal">Cancelar</button>
253           <button class="btn" disabled id="envio_button">Aceptar</button>
254         </div>
255       </form>
256     </div>
257   </div>
258 </div>

```

Figura 4.75 Estructura HTML del modal para agregar un nuevo servicio a la DB.

Este es el formulario que se encuentra dentro del modal de agregar nuevo Servicio, este modal es uno de los más extensos ya que contiene muchos datos, esta seccionado en tres partes, que son los datos del cliente, datos del vehículo y datos del servicio. Solo son campos que se debe de llenar el usuario.

```

function activar_envio(){

    if (document.getElementById("nombre").value!=""&& document.getElementById("cp").value!=""&& document.getElementById("estado").value!=""&&
        document.getElementById("colonia").value!=""&& document.getElementById("domicilio").value!=""&& document.getElementById("telefono").value!=""&& document.getElementById("correo").value!="") {

        var aux=j-contador;
        console.log(aux);
        for (var i = 0; i < j; i++) {
            try{

                if (document.getElementById("id_vehiculo"+i).value!="" && document.getElementById("nombre_vehiculo"+i).value!=""&& document.
                    getElementById("funcion"+i).value!="" && document.getElementById("matricula"+i).value!="" && document.getElementById("fecha_
                    inicio"+i).value!="" && document.getElementById("fecha_termino"+i).value!="" && document.getElementById("horas_trabajo
                    "+i).value>0 && document.getElementById("mat_horas"+i).value>0 && document.getElementById("no_mantenimientos"+i).value>0) {

                    document.getElementById("envio_button").classList.add("btn-success");
                    document.getElementById("envio_button").disabled = false;

                } else {
                    document.getElementById("envio_button").classList.remove("btn-success");
                    document.getElementById("envio_button").disabled = true;
                    break;
                }

            }catch (TypeError) {
                console.log("no existe ese objeto con ese nombre");
            }
        }

    } else{
        document.getElementById("envio_button").classList.remove("btn-success");
        document.getElementById("envio_button").disabled = true;
    }
}

```

Figura 4.75 Función de JavaScript que evalúa si todos los campos han sido llenados para activar el envío.

Esta función es propia del modal de agregar servicio, verifica que todos los campos estén llenos para evitar envío de campos vacíos, se activa el botón si todos tienen texto.

```

function llenar_select(data_service,indicemax){

    var indice_select=(indicemax-1);
    var datos_servicio_pasado=data_service;
    console.log(data_service);
    console.log(indicemax);
    for (var t = 0; t <= indice_select; t++) {
        $("#" + "id_vehiculo_edit" + t).empty();
        //console.log(j);
        $("#" + "id_vehiculo_edit" + t).append('<option value="" disabled selected>.:Selecciona.:</option>');
    }

    $.ajax({}).done(function(vehiculos_filtrado_edit) {
        console.log(vehiculos_filtrado_edit);
        console.log("hola");
        if(vehiculos_filtrado_edit!=null){

            for (var t = 0; t <= indice_select; t++) {

                //vehiculos sin usar
                if(vehiculos_filtrado_edit[0] !=null){
                    for(var i=0;i<vehiculos_filtrado_edit[0].length;i++){

                        $("#" + "id_vehiculo_edit" + t).append('<option value="'+vehiculos_filtrado_edit[0][i].id+'">' + vehiculos_filtrado_edit[0][i].matricula + '</option>');
                    }
                }
                //vehiculos con fecha terminada
                if(vehiculos_filtrado_edit[1] != null){ }

                if(vehiculos_filtrado_edit[2] != null){ }

                console.log("valor del for"+t);

            }

            for (var i = 0; i <= indice_select; i++) {
                console.log("id_vehiculo_edit"+i);
                $("#" + "id_vehiculo_edit" + i).val(datos_servicio_pasado[i].id_vehiculo);

            }
        }
    });
}

```

Figura 4.76 Función de JavaScript que se encarga de llenar los campos “select” con sus respectivos datos.

Esta función de JavaScript se encarga de llenar los “select”, son los campos de opciones de los vehículos, se llenan atreves de Ajax. Las búsquedas rápidas las hace por el controlador de “SearchesController” de la misma forma que se hizo en el módulo de “Usuarios” para llenar los datos del modal editar, solo que aquí llenamos el “select” de opciones, se hizo de esa forma ya que los campos son dinámicos y no se sabe cuántos agregara el usuario.

Se insertará código HTML mediante otra función se activa por el botón de “agregar vehículo”, este se encargará de agregar más campos según así lo desee el usuario, esta es una de las funciones más complejas del sistema ya que se deben de agregar más campos justo al momento que el usuario está agregando un nuevo registro. De esta función es que las demás funciones nacen ya que se deben de llenar cosas a nuevos campos agregados.

```
1190
1191 var j=1;
1192 var contador=0;
1193 function agregar_prueba(){
1194
1195     if(j>0){
1196
1197         $("#contenedor_padre").append(
1198
1199             '<div id="contenedor_hijo'+j+'">'+'  

1200             '</div>'  

1201
1202     );  

1203
1204     /*  

1205     $.ajax({  

1206         done(function(vehiculos_services) {  

1207             });  

1208         }  

1209
1210         j++;  

1211         contador++;  

1212
1213         $("#id_vehiculo"+(j-1)).empty();
1214         console.log(j);
1215         $("#id_vehiculo"+(j-1)).append('<option value="" disabled selected>.:Selecciona:</option>');
1216         $.ajax({  

1217             done(function(vehiculos_filtrado) {
1218                 if(vehiculos_filtrado!=null){  

1219
1220                     //vehiculos sin usar
1221                     if (vehiculos_filtrado[0] != null){  

1222                         //vehiculos con fecha terminada
1223                         if (vehiculos_filtrado[1] != null){  

1224
1225                             //console.log(vehiculos_filtrado[0]);
1226                             //console.log(vehiculos_filtrado[1]);
1227
1228                         }
1229
1230                     }
1231
1232                 }
1233             });
1234
1235             //console.log(j);
1236             //console.log(contador);
1237             document.getElementById("cantidad_vehiculos").value=j;
1238             activar_envio();
1239
1240     });
1241
1242     }
1243
1244 }
```

Figura 4.77 Función de JavaScript para agregar campos dinámicos al modal mediante un botón.

Mediante esta función de JavaScript es donde se agregar código HTML a la página, esto con el fin de agregar más campos al modal de agregar nuevo servicio, este permite al usuario agregar más vehículos al formulario.

```
1505     $(document).on('click', '.eliminar_hijo', function(){
1506         var id=$(this).attr("id");
1507         $('#contenedor_hijo'+id+'').remove();
1508         contador--;
1509         //j--;
1510         document.getElementById("cantidad_vehiculos").value=contador;
1511         activar_envio();
1512     });
1513 }
```

Figura 4.78 Función de JavaScript que permite la eliminación de los campos dinámicos en el modal.

Esta función sirve para eliminar campos agregados dinámicamente se activa mediante el botón de “eliminar”, elimina los campos agregados con la función código HTML.

```

978
979     function buscar_cp(cp){
980
981         $.ajax({
982             url: "{{url('/get_cp')}}"+'/' +cp,
983             dataType: "json",
984             //context: document.body
985         }).done(function(cp_data) {
986             document.getElementById("estado").value=null;
987             $("#colonia").empty();
988             if(cp_data==null){
989                 document.getElementById("estado").value=null;
990                 $("#colonia").empty();
991             }else{
992                 document.getElementById("estado").value=cp_data[0].estado;
993                 for(var i=0;i<cp_data.length;i++){
994                     $("#colonia").append('<option value="'+cp_data[i].colonia+'">' +cp_data[i].colonia+'');
995                 }
996             }
997         });
998
999     }
1000
1001 //esta es la misma de que otra pero esta es para la de editar
1002

```

Figura 4.79 Función de JavaScript que se encarga de hacer una búsqueda a la DB mediante el campo CP.

Esta función tiene como propósito buscar el código postal “cp” que agrega el usuario para poder darle las colonias en el modal de agregar servicio, solo agrega opciones dependiendo el código postal agregado, de la misma forma se usa Ajax para poder hacer el llenado mediante el controlador de “SearchesController” que se encarga de las búsquedas rápidas a la base de datos, hace un filtro a la tabla “cat_estados” que es donde están todas las colonias según el código postal.

```

1134 ▼    function diferencia_dias(indice){
1135        var fecha_1= new Date(document.getElementById("fecha_inicio"+indice).value).getTime();
1136        var fecha_2= new Date(document.getElementById("fecha_termino"+indice).value).getTime();
1137        //alert(fecha_1);
1138        //alert((fecha_2-fecha_1)/(1000*60*60*24)); //La multiplicacion te da los milisegundos que tiene un dia, y el metodo getTime en milisegundos.
1139        var result=((fecha_2-fecha_1)/(1000*60*60*24))*24;
1140        if (result>=1){
1141            document.getElementById("horas_trabajo"+indice).value=((fecha_2-fecha_1)/(1000*60*60*24))*24;
1142        }else{
1143            document.getElementById("horas_trabajo"+indice).value=null;
1144            document.getElementById("horas_trabajo"+indice).placeholder="operacion fallida";
1145        }
1146    }
1147

```

Figura 4.80 Función de JavaScript que se encarga de hacer la operación de las horas de trabajo.

Esta función se encarga de darle la diferencia entre las fechas de los días de los campos de “fecha inicio” y “fecha término” para así determinar cuántos días hábiles tendrá de trabajo el vehículo en dicho servicio.

```

1162     function mantenimiento_horas(indice){
1163
1164         var dato= document.getElementById("mat_horas"+indice).value;
1165         var dato_2=0;
1166         if (document.getElementById("horas_trabajo"+indice).value>=1 && dato>=1){
1167             dato_2= document.getElementById("horas_trabajo"+indice).value;
1168             document.getElementById("no_mantenimientos"+indice).value=Math.round(dato_2/dato);
1169         }else{
1170             document.getElementById("no_mantenimientos"+indice).value=null;
1171             document.getElementById("no_mantenimientos"+indice).placeholder="operacion fallida";
1172         }
1173         activar_envio();
1174     }

```

Figura 4.81 Función de JavaScript que realiza la operación de numero de mantenimientos por vehículo.

esta función determina el número de mantenimiento que tendrá el vehículo en dicho servicio, toma como datos los campos de “Horas de Trabajo” y “Mant. después de (Hrs)”. Y así determina el número de mantenimientos que tendrá durante el servicio.

Datos del Cliente

Nombre / Empresa	C.P	Estado	Colonia
ADRIANA	56500	Estado de México	San Salvador Tecamachalco

Domicilio

5 de mayo mz1 lt 6	Telefono	Correo
	5521439037	adriana@gmail.com

Datos del Vehículo

Matrícula del Vehículo	Nombre	Función	Agregar Vehículo
qwe-123-ew	camion	traer tierra	Agregar Vehículo

Datos del Servicio

Matrícula del Vehículo	Fecha de inicio	Fecha de Término	Horas de Trabajo
qwe-123-ew	01/11/2022	03/12/2022	768

Mant. después de (Hrs)	No. Mantenimientos	Observaciones
15	51	ninguna

Datos del Vehículo

Matrícula del Vehículo	Nombre	Función	eliminar
54fd-tr	tractor	quitar tierra	eliminar

Observaciones Generales

Figura 4.82 Vista del modal de agregar un nuevo servicio situado en el módulo de “Servicios”.

Y así es como es como se interactúa con el modal, se agrega más campos que en este caso el usuario entiende que es agregar otro vehículo, o se eliminan si así fuera el

caso, pero esto trabaja mediante los datos agregados en “Catálogos”. Están presentes los botones que agregan y eliminan los campos dinámicos que el usuario agregue, no tiene límite de campos.

```

    public function agregar_service(Request $request){
}
try {
    DB::table("servicios")->insert([
]);
//obtenemos el id del registro servicios para ligarlo con los vehiculos agregados
$id = DB::getPdo()->lastInsertId();
$contador=0;
//este es para que se cambie y se agregue como nuevo
for ($i=0;$i<=$request['cantidad_vehiculos'];$i++) {
}

for ($i=0;$i<=$request['cantidad_vehiculos'];$i++) {
    if(isset($request["id_vehiculo"][$i])){
        DB::table("servicio_vehiculo")->insert([
            "id_servicio"=>$id,
            "id_vehiculo"=>$request["id_vehiculo"][$i],
            "nombre_vehiculo"=>$request["nombre_vehiculo"][$i],
            "funcion_vehiculo"=>$request["funcion"][$i],
            "matricula_vehiculo"=>$request["matricula"][$i],
            "fecha_inicio"=>$request["fecha_inicio"][$i],
            "fecha_fin"=>$request["fecha_termino"][$i],
            "horas_trabajo"=>$request["horas_trabajo"][$i],
            "mantenimiento_h"=>$request["mat_horas"][$i],
            "no_mantenimientos"=>$request["no_mantenimientos"][$i],
            "observaciones"=>$request["observaciones"][$i],
        ]);
        $contador++;
    }
}
for ($i=0;$i<=$request['cantidad_vehiculos'];$i++) {
}

$data =[ "usuario" => Auth::user()->sname,"empresa" => $request["nombre"],"direccion" => $request["domicilio"],"cantidad" => $contador,
        "id_servicio" => $id,"tipo" => "nuevo"];
Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Servicio Creado",$data,"service"));

return redirect()->back()->with(['message' => "Se Guardo correctamente el Servicio", 'color' => 'success']);
}

```

Figura 4.83 Método de PHP situado en el controlador “ServicesController”, agregar un nuevo servicio a la DB.

Este es el método que agrega el registro de un nuevo servicio, se agrega el servicio y también se agregan los vehículos agregados a ese servicio a otra tabla que conecta a los servicios con los vehículos, una vez que agrega el registro regresa a la vista anterior con un mensaje de éxito o de error según sea el caso, de igual manera enviamos un correo de alerta para que el administrador este enterado de lo que pasa en el sistema.



Figura 4.84 Vista del correo electrónico al crear un nuevo servicio en el sistema.

Este es el correo que envía el sistema, en este caso fue en el módulo “Servicios”, este formato será similar para los demás movimientos dentro de este módulo.

Servicios						
Se Guardo correctamente el Servicio						
<input type="button" value="Agregar"/> Mostrar <input type="button" value="10"/> registros <input type="text" value="Buscar:"/> <input type="button" value=""/>						
Folio	No. Vehículos	Cliente	F. Inicio	F. Termino	Estatus	
CMZ-1	2	ADRIANA	2022-11-01	2022-12-03	espera	

Mostrando registros del 1 al 1 de un total de 1 registros

Figura 4.85 Vista del Mensaje de éxito al agregar un nuevo servicio a la DB.

Para poder editar el registro se debe de elegir el registro y seleccionar la opción de editar, el menú despegable de la tabla es el mismo que se viene usando desde el módulo “Usuarios” que contiene la opción de editar y eliminar.

```

786
787     function editar_registro(){
788         $("#colonia_edit").empty();
789         $("#contenedor_padre_edit").empty();
790         $("#observaciones_edit_edit").empty();
791         j_2=1;
792         var id_seleccion_vehiculo=0;
793         var id_seleccion_vehiculo_2=0;
794         var valor;
795         contador_2=0;
796         $.ajax({
797             url: "{{url('/get_servicios')}}"+'/'+id_servicio,
798             dataType: "json",
799             //context: document.body
800         }).done(function(service_data) {
801             if(service_data==null){
802                 document.getElementById("folio_edit").value=null
803                 document.getElementById("nombre_edit").value=null;
804                 document.getElementById("cp_edit").value=null;
805                 document.getElementById("estado_edit").value=null;
806                 $("#colonia_edit").empty();
807                 document.getElementById("domicilio_edit").value=null;
808                 document.getElementById("telefono_edit").value=null;
809                 document.getElementById("correo_edit").value=null;
810                 document.getElementById("id_servicio_edit").value=null;
811                 $("#observaciones_edit_edit").empty();
812             }else{
813                 document.getElementById("folio_edit").value="CMZ-"+service_data.id;
814                 document.getElementById("nombre_edit").value=service_data.nombre;
815                 document.getElementById("cp_edit").value=service_data.cp;
816                 document.getElementById("estado_edit").value=service_data.estado;
817                 $("#colonia_edit").append('<option value="'+service_data.colonia+'">' + service_data.colonia + '</option>');
818                 document.getElementById("domicilio_edit").value=service_data.domicilio;
819                 document.getElementById("telefono_edit").value=service_data.telefono;
820                 document.getElementById("correo_edit").value=service_data.correo;
821                 document.getElementById("id_servicio_edit").value=service_data.id;
822                 $("#observaciones_edit_edit").append(service_data.observaciones);
823             $.ajax({
824                 url: "{{url('/get_servicio_vehiculos')}}"+'/'+service_data.id,
825                 dataType: "json",
826                 //context: document.body
827             }).done(function(servicio_vehiculos_data) {
828
829                 if(servicio_vehiculos_data==null){
830
831                     $("#contenedor_padre_edit").empty();
832                     j_2=1;
833                     contador_2=0;
834
835                 }else{
836
837                     for(var i=0;i<servicio_vehiculos_data.length;i++){

```

Figura 4.86 función de JavaScript que llena los datos del modal editar servicio según sea el registro seleccionado.

Para el llenado de los datos del modal editar se usó la misma técnica de Ajax para llenar los campos del registro seleccionado, hace las búsquedas rápidas por el controlador de “SearchesController”, recordemos que el modal de editar registro es muy similar que el de agregar servicio, pero solo se cambiaron etiquetas, las funciones que le dan vida al modal de agregar servicio son las mismas para el modal de editar ya que es exactamente lo mismo, pero en vez de agregar se edita.

Editar Servicio (X)

Folio			
CMZ-1			
Datos del Cliente			
Nombre / Empresa	C.P	Estado	Colonia
ELVIRA	34500	Durango	Canelas
Domicilio	Telefono	Correo	
5 de mayo mz1 lt 6	5521439037	adriana@gmail.com	
Datos del Vehiculo			
Matricula del Vehiculo	Nombre	Funcion	
qwe-123-ew - p	camion	traer tierra	Agregar Vehiculo
Datos del Servicio			
Matricula del Vehiculo	Fecha de inicio	Fecha de Termino	Horas de Trabajo
qwe-123-ew	01/11/2022	03/12/2022	768
Mant. después de (Hrs)	No. Mantenimientos	Observaciones	
15	51	ninguna	
Observaciones Generales			
ninguna			
		Cancelar	Actualizar

Figura 4.87 Vista del Modal de editar un servicio, situado en el módulo “Servicios”.

Se quito un vehículo y se cambió el nombre del cliente junto con el de “cp”, para actualizar los datos de un servicio a manera de ejemplo, los vehículos seleccionados se eliminan los que tenía y se agregan de nuevo, esto con el fin de agilizar el tiempo de carga.

```

public function actualizar_service(Request $request){
    if ($request["id_servicio_edit"]==null || $request["id_servicio_edit"]==0) {
        return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
    }
    try {
        DB::table("servicios")->where("id",$request["id_servicio_edit"])->update([
            ...
        ]);
        DB::table("servicio_vehiculo")->where("id_servicio",$request["id_servicio_edit"])->delete();
        DB::table("servicio_vehiculo_registro")->where("id_servicio",$request["id_servicio_edit"])->delete();

        $contador=0;
        for ($i=0;$i<=$request['cantidad_vehiculos'];$i++) {
            ...
        }
        for ($i=0;$i<=$request['cantidad_vehiculos'];$i++) {

            if(isset($request["id_vehiculo"][$i])){
                ...
            }

            for ($i=0;$i<=$request['cantidad_vehiculos'];$i++) {
                ...
            }
        }
        $data =[ "usuario" => Auth::user()->name, "empresa" => $request["nombre"], "direccion" => $request["domicilio"], "cantidad" => $contador,
            "id_servicio" => $request["id_servicio_edit"], "tipo" => "actualizar"];
        Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Servicio Actualizado",$data,"service"));

        return redirect()->back()->with(['message' => "Se Actualizo correctamente el Servicio", 'color' => 'warning']);
        //echo "seguindo";
    } catch (\Exception $e) {
        return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
    }
}

```

Figura 4.88 Método de PHP situado en el controlador “ServicesController” que permite editar un servicio en la DB.

Es lo mismo que hace el método de agregar, pero en este caso indicamos que se debe de actualizar el registro con los datos enviados por el formulario del modal editar y de la misma forma regresa a la vista anterior con un mensaje de éxito o de error, según sea el caso, también se envía un correo de alerta de que se actualizo un registro.

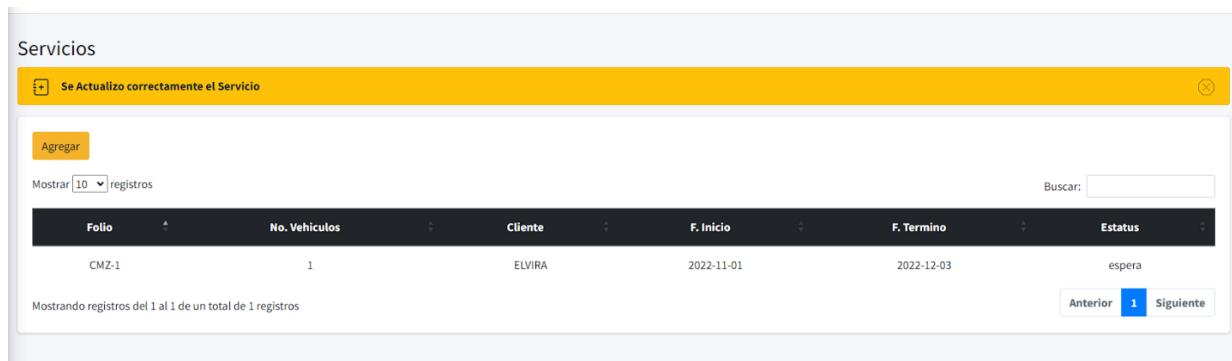


Figura 4.89 Vista del mensaje de alerta al editar un servicio en el módulo de “Servicios”.

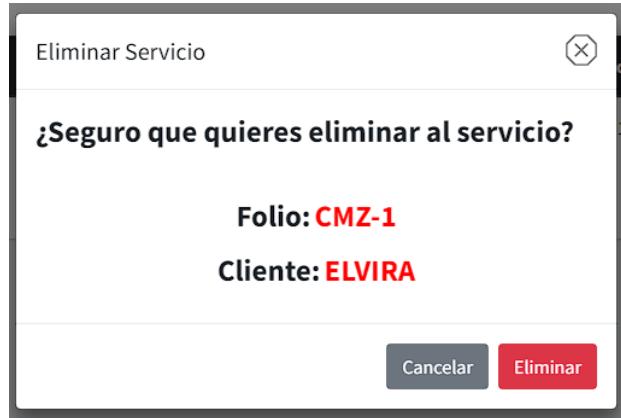


Figura 4.90 Vista del modal eliminar servicio situado en el módulo “Servicios”.

Este es el modal de eliminar, para eliminar un servicio de la base de datos solo es necesario el “id” del registro, se pone esta alerta para que el usuario este de acuerdo con el registro a eliminar de la base de datos definitivamente, el “id” se obtiene cuando se selecciona el registro y se muestra el menú despegable.

```

230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255

    public function eliminar_service(Request $request){
        try {
            if ($request["id_servicio_edit"]!=0) {
                $service_data=DB::table("servicios")->where("id",$request["id_servicio_edit"])->first();
                DB::table("servicios")->where("id",$request["id_servicio_edit"])->delete();

                $data =["usuario" => Auth::user()->name,"empresa" => $service_data->nombre,"direccion" => $service_data->domicilio,"cantidad" => "0",
                        "id_servicio" => $service_data->id,"tipo" => "eliminar"];
                Mail::to("nuckelavjee95@gmail.com")->send(new MessageSent("Servicio Eliminado",$data,"service"));

                DB::table("servicios")->where("id",$request["id_servicio_edit"])->delete();
            }
            return redirect()->back()->with(['message' => "Se Elimino correctamente el Servicio", 'color' => 'danger']);
        } catch (\Exception $e) {
            return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
        }
    }
}

```

Figura 4.91 Método de PHP situado en el controlador “ServicesController”, permite eliminar un servicio de la DB.

Este método es el encargado de eliminar el registro de la base de datos, en este caso elimina definitivamente el registro del servicio, una vez eliminado el registro del servicio también se eliminan la vinculación con los vehículos seleccionados, esto lo hace de forma automática la base de datos mediante la “foreign key”. De la misma manera enviamos un correo de alerta de un registro fue eliminado.



Figura 4.92 Vista del mensaje de alerta al eliminar un servicio de la DB en el módulo “Servicios”.

Las rutas del módulo de “Servicios” son las siguientes.

```

54     //services
55     Route::get('/Servicios',[ServicesController::class,'view_services'])->name('services');
56     Route::post('/guardar_servicio',[ServicesController::class,'agregar_service'])->name('agregar_service');
57     Route::post('/Actualizar_servicio',[ServicesController::class,'actualizar_service'])->name('actualizar_service');
58     Route::delete('/Eliminar_servicio',[ServicesController::class,'eliminar_service'])->name('eliminar_service');
59

```

Figura 4.93 Rutas que activan los métodos del controlador “ServicesController”.

Cada ruta tiene la “url” por la cual el usuario final hará consulta y activa el método correspondiente de la clase “ServicesController” que es el controlador de este módulo.

SP3 H3 T11 REPORTES.

Para la creación de reportes en PDF, se usará una librería que trabaja muy bien con Laravel, DOMPDF es una librería para la creación de PDF muy buena y que cuenta documentación, permite crear PDF a partir de una vista o código HTML.

Instalación de Laravel mediante terminal “Cmder” de Laragon.

```

Discovered package: laravelcollective/html
Package manifest generated successfully.
80 packages you are using are looking for funding.
Use the "composer fund" command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force
No publishable resources for tag [laravel-assets].
Publishing complete.

  Alternatively, run:
C:\laragon\www\matzy_system>php artisan vendor:publish --provider="Barryvdh\DomPDF\ServiceProvider"
Copied File [\vendor\barryvdh\laravel-dompdf\config\dompdf.php] To [\config\dompdf.php]
Publishing complete.

C:\laragon\www\matzy_system>

```

Figura 4.94 Instalación de la librería DOMPDF mediante Cmder.

Una vez que termine de agregar DOMPDF al proyecto, tendremos que crear una vista que será la que renderizara la librería y mostrara como resultado el PDF.

```

256     public function pdf_service($id){
257
258         $datos=DB::table("servicios")->where("id",$id)->first();
259         $servicio_vehiculos=DB::table("servicio_vehiculo")->where("id_servicio",$id)->get();
260
261         $pdf = PDF::loadView('Services.pdf_services',compact("datos","servicio_vehiculos"))->setPaper(array(0,0,1186,1536));
262         //\$nombre_pdf="Matriz Master_".\$proyectos->nOMBRE.".pdf";
263
264         return $pdf->stream("folio_CMZ-".$id.".pdf");
265     }

```

Figura 4.95 Método de PHP situada en el controlador “ServicesController”, retorna el PDF con los datos.

Se agrego otra funcion al controlador de “ServicesController”, es para que el usuario pueda visualizar el PDF del servicio creado, el PDF contendra todos los datos del servicio asi como los vehiculos seleccionados o agregados, el envio de datos es la misma que cuando se retorna una vista normal, solo que aqui se ocupa el metodo de “loadView” de DOMPDF que indica que le pasaremos una vista o HTML.

```

51   <body>
52
53   <header>
54     <p style=" margin-top: 55px; padding-left: 1220px; position: absolute; font-size: 30px;">Folio: CMZ-{{$datos->id}}</p>
55     <p style=" margin-top: 110px; padding-left: 50px; position: absolute; font-size: 30px;">Datos del Cliente</p>
56
57     <table class="table" style="border-collapse:separate; width: 100%; margin-top: 145px; border-spacing:0 55px;">
58
59       <tbody> ==|
60
61       </tbody>
62     </table>
63
64     <table class="table" style="border-collapse:separate; width: 100%; margin-top: -60px; border-spacing:0 55px;">
65
66       <tbody>
67         <tr style="font-weight: 0;"> ==
68
69         @foreach($servicio_vehiculos as $servicio_vehiculo)
70           <tr style="font-weight: 0;">
71             <td style="font-size: 30px; "></td>
72             <td style="font-size: 30px; ">Matricula: {{$servicio_vehiculo->matricula_vehiculo}}</td>
73             <td style="font-size: 30px; ">Nombre: {{$servicio_vehiculo->nOMBRE_vehiculo}}</td>
74             <td style="font-size: 30px; " colspan="2">Funcion: {{$servicio_vehiculo->funcion_vehiculo}}</td>
75             <td style="font-size: 30px; width: 50px;"></td>
76
77           </tr>
78           <tr style="font-weight: 0;">
79             <td style="font-size: 30px; width: 50px;"></td>
80             <td style="font-size: 30px; width: 580px; colspan="2">Fecha de inicio del Servicio del Vehiculo: {{$servicio_vehiculo->fecha_inicio}}</td>
81             <td style="font-size: 30px; " colspan="2">Fecha de termino del Servicio del Vehiculo: {{$servicio_vehiculo->fecha_fin}}</td>
82             <td style="font-size: 30px; width: 50px;"></td>
83
84           </tr>
85           <tr style="font-weight: 0;">
86             <td style="font-size: 30px; "></td>
87             <td style="font-size: 30px; " colspan="2">Horas de servicio para el Vehiculo: {{$servicio_vehiculo->mantenimiento_h}}hrs.</td>
88             <td style="font-size: 30px; " colspan="2">No. Mantenimientos para el vehiculo en el periodo del servicio: {{$servicio_vehiculo->no_mantenimientos}}</td>
89             <td style="font-size: 30px; width: 50px;"></td>
90
91           </tr>
92           <tr style="font-weight: 0;">
93             <td style="font-size: 30px; "></td>
94             <td style="font-size: 30px; " colspan="2">Observaciones:<br>{{$servicio_vehiculo->observaciones}}</td>
95             <td style="font-size: 30px; width: 50px;"></td>
96
97           </tr>
98         @endforeach
99         <tr style="font-weight: 0;"> ==
100        </tr>
101      </tbody>
102    </table>
103  </header>
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121

```

Figura 4.96 Estructura HTML del PDF para el módulo “Servicios”.

Esta es la vista que renderizara DOMPDF, solo es acomodo de texto y dispersión del mismo en la vista. Se utilizo tablas ya que ayudan mucho al acomodo de texto cuando se

trata de texto dinámico, el diseño del PDF es acorde al diseño en general del sistema, para poder visualizar el PDF se agregó una opción más al menú despegable de la tabla que activa un modal que contiene es la vista renderizada en PDF.

```
<!--menu de opciones de la tabla-->
<div id="menu_opciones" class="visible_off" style="padding: 20px; background-color: #858585bd;">

    <button type="button" class="close" style="margin-right: -17px; margin-top: -20px;" onclick="cerrar_menu();">
        <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor" class="bi bi-x-octagon" viewBox="0 0 16 16" >
            </svg>
    </button>
    <button class="btn btn-danger" style="margin-bottom: 10px; font-weight: bold; margin-top: 10px; width: 100%;" data-toggle="modal" data-target="#eliminar_registro" onclick="eliminar_registro();">
        <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor" class="bi bi-pencil-square" viewBox="0 0 16 16" >
            </svg>
        Eliminar
    </button>
    <br>
    <button class="btn btn-warning" style="margin-bottom: 10px; font-weight: bold; width: 100%;" data-toggle="modal" data-target="#editar_registro" onclick="editar_registro();">
        <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor" class="bi bi-pencil-square" viewBox="0 0 16 16" >
            <path d="M15.502 1.94a.5.5 0 0 1 .706l14.459 3.691-2-2L13.502.646a.5.5 0 0 1 .707 0l1.293 1.293zm-1.75 2.456-2-2L4.939 9.21a.5.5 0 0 1 .121.196l-.805 2.414a.25.25 0 0 0 .316.316l2.414-.805a.5.5 0 0 1 -.196-.1216.813-6.814z"/>
            <path fill-rule="evenodd" d="M13.5A1.5 1.5 0 0 0 2.5 15h1a1.5 1.5 0 0 0 1.5-1.5v-6a.5.5 0 0 1 -.5h-11a.5.5 0 0 1 -.5-.5v-11a.5.5 0 0 1 -.5-.5H9a.5.5 0 0 0 0-1H2.5A1.5 1.5 0 0 0 0 1.5v11z"/>
        </svg>
        Editar
    </button>
    <br>
    <button class="btn btn-info" style="margin-bottom: 10px; font-weight: bold; width: 100%;" data-toggle="modal" data-target="#pdf" onclick="agregar_url();">
        <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor" class="bi bi-filetype-pdf" viewBox="0 0 16 16" >
            </svg>
        PDF
    </button>
</div>
```

Figura 4.97 Estructura HTML del menú despegable de la tabla para el módulo “Servicios”.

Aquí es la parte del menú despegable de menú despegable de la tabla, solo se agregó un botón más al menú que es el de PDF, este botón activara el botón que muestra el PDF.

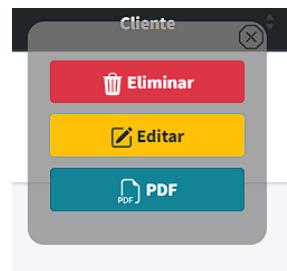


Figura 4.98 Vista del menú despegable de la tabla para el módulo “servicios”.

Así es como luce ahora el menú despegable de la tabla, este nuevo botón va acompañado de una función de JavaScript que busca el registro de la tabla y mostrar el PDF referente a ese registro, esto lo hace cambiando la “url” de la etiqueta que permite visualizar PDF en HTML.

```

627     function agregar_url() {
628         var url_server = "{{url('/PDF_servicio')}}"/";
629         document.getElementById('visor_pdf').src=url_server+id_servicio;
630         document.getElementById("ir_otro_lado").href=url_server+id_servicio;
631     }

```

Figura 4.99 Función de JavaScript para agregar la url al modal para visualizar el PDF renderizado.

Solo agrega el “id” del registro seleccionado a la “url” del componente que permite visualizar el PDF.

```

<!-- modal pdf-->
<div class="modal fade" id="pdf" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog modal-xl modal-dialog-centered">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title" id="exampleModalLabel">Ver PDF</h5>
                <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                    <svg xmlns="http://www.w3.org/2000/svg" width="26" height="26" fill="currentColor" class="bi bi-x-octagon" viewBox="0 0 16 16" style="color: #333;>
                        <path d="M4.54.146a.5.5 0 0 1 4.893 .0h6.214a.5.5 0 0 1 .353.146l4.394 4.394a.5.5 0 0 1 .146.353v6.214a.5.5 0 0 1 .146.353l4.54.146zM5.1 4.394a.5.5 0 0 1 -.353.146h4.893a.5.5 0 0 1 -.353-.146L146 11.46a.5.5 0 0 1 0 11.107V4.893a.5.5 0 0 1 .146-.353L4.54.146zM5.1 5.1v5.8l1.5 1 15h5.814l1.4-1.4V5.1110.9 1H5.1z"/>
                    </svg>
                </button>
            </div>
            <div class="modal-body">
                <div class="col-md-12" style="text-align: center; display: none;" id="no_se_mira">
                    <p>Ups! &nbsp;&nbsp; !CREO QUE NO SE VE BIEN EL PDF, VAMOS A OTRA PAGINA OK!</p>
                    <a class="btn btn-success" target="" href="#" id="ir_otro_lado">VAMOS! <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-box-arrow-up-right" viewBox="0 0 16 16">
                        <path fill-rule="evenodd" d="M8.636 3.5a.5.5 0 0 1 1.272 0l1.493 1.493a.5.5 0 0 1 0 1.0l-1.493 1.493a.5.5 0 0 1 -1.272 0l-1.493 -1.493a.5.5 0 0 1 0 -1.0l1.493 -1.493a.5.5 0 0 1 1.272 0z"/>
                    </svg>
                </a>
                <embed type="application/pdf" src="" style="width:100%; height: 600px;" id="visor_pdf">
                </embed>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-success" data-dismiss="modal">Aceptar</button>
            </div>
        </div>
    </div>
</div>

```

Figura 4.100 estructura HTML del modal para visualizar el PDF para el módulo “Servicios”.

Este es el modal que muestra el PDF, lo único que se agregó en este modal es una etiqueta que permite mostrar PDF en HTML o en un navegador, la etiqueta es “embed”, está el a la que se le cambia la “url” según sea el registro.

```

60     Route::get('/PDF_servicio/{id}',[ServicesController::class, 'pdf_service'])->name('pdf_service');
61

```

Figura 4.101 Ruta para visualizar el PDF del módulo “Servicios”.

Esta es la ruta que activa el método para mostrar el PDF, pertenece al controlador “ServicesController”, pide un parámetro que en este caso es el “id” del registro y así poder hacer un filtrado en la base de datos a base de ese “id”.

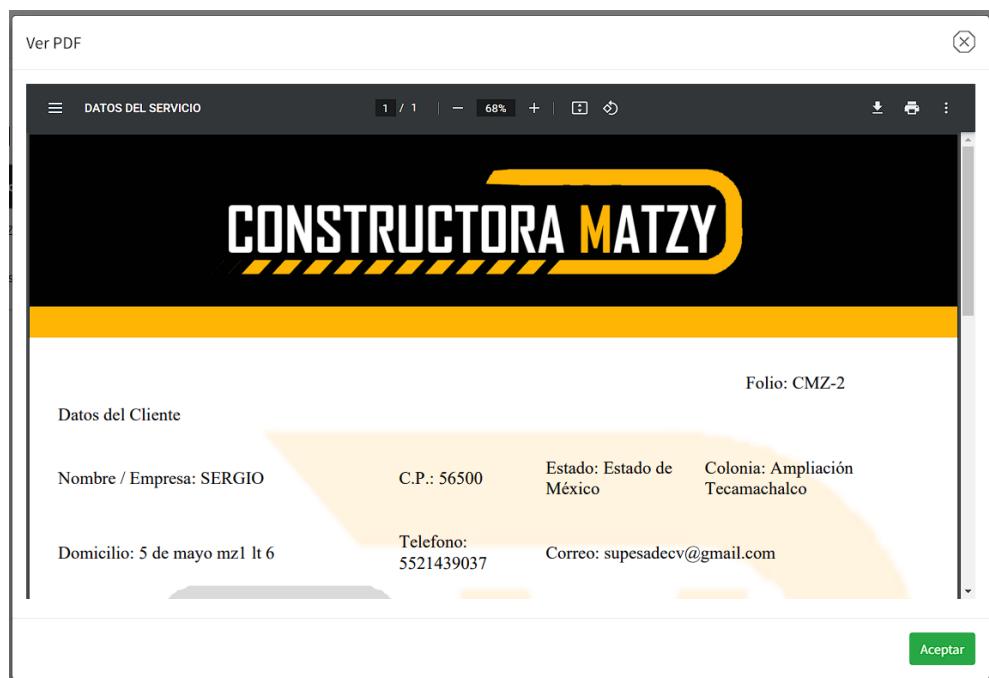


Figura 4.102 Vista del modal para visualizar el PDF del módulo “Servicios”.

Así es como se muestra el PDF del servicio, el reporte contiene todos los datos del servicio creado, permite descargarlo, imprimirlo las funciones que encontramos al visualizar un PDF en el navegador. Esta forma es más útil que mandar al usuario a otra vista, muestra el PDF de una forma rápida para poder imprimirlo o descargarlo sin salir del sistema, de igual manera para poder generar los PDF debes de iniciar sesión, esto para evitar filtración de información y así se garantiza la integridad de los datos de la empresa.

SP3 H3 LIBERACIÓN Y ENTREGA DEL DEPARTAMENTO DE SERVICIOS.

Entrega de Sprint

Información de la empresa y proyecto:

Empresa/ Organización	Constructora Matzy S.A de C.V.
Proyecto	Matzy System

información del sprint:

Iteración/Sprint	ID	Enunciado de la historia
SP3 Departamento de servicios	H3	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita hacer la toma de datos necesarios por parte del cliente y la disponibilidad de mis vehículos y fechas en las que estos estarán disponibles y de esta manera, con la finalidad de que pueda decidir correctamente cuando iniciar un servicio, obteniendo un reporte interno con todo lo registrado y "modulo" que me permita registrar mis vehículos, con la finalidad de que no tenga que registrarlos cada vez.

Información de la entrega:

F	
Numero de iteración/ Sprint	SP3 H3/Departamento de servicios
Persona responsable del recibimiento	
Cargo	Dueño de la empresa

información del equipo de trabajo:

Nombre	Rol
Raul Guerrero Ramírez	Scrum master
Antonio Juárez Andrade	Equipo de trabajo

Observaciones:

Bajo el principio de Lean, sujeto a la o las demandas del cliente o dueño de la empresa, el sprint fue entregado con cada uno de los requerimientos citados en el enunciado de la historia, dando por concluido al cien por ciento el total de las tareas impuestas para el sprint SP3 H3 Departamento de servicios

Firma de conformidad del encargado
de recibir el sprint

Representante legal.
"Constructora Matzy"

Figura 4.103 Hoja de liberación del sprint “departamento de servicios”.

TABLERO DE TAREAS FINAL DEL SP3 H3.

Historia de usuario	Tareas	En proceso	Concluido
H3 Departamento de servicios	SP3 H3 T1		T1
H3 Departamento de servicios	SP3 H3 T2		T2
H3 Departamento de servicios	SP3 H3 T3		T3
H3 Departamento de servicios	SP3 H3 T4		T4
H3 Departamento de servicios	SP3 H3 T5		T5
H3 Departamento de servicios	SP3 H3 T6		T6
H3 Departamento de servicios	SP3 H3 T7		T7
H3 Departamento de servicios	SP3 H3 T8		T8
H3 Departamento de servicios	SP3 H3 T9		T9
H3 Departamento de servicios	SP3 H3 T10		T10
H3 Departamento de servicios	SP3 H3 T11		T11
Porcentaje de avance			
0-20%	21-40%	41-60%	61-80%
			T1,T2,T3,T4, T5,T6,T7,T8, T9,T10,T11

SPRINT BACKLOG SP4 H4

ID	Product backlog		
H4	Yo, “representante legal de Constructora Matzy”, necesito un “módulo” que me permita asignar un presupuesto a un servicio que ya se encuentra iniciado, para poder registrar mis ingresos y gastos correspondientes a este servicio con el cliente de turno y así obtener las ganancias, obteniendo un informe interno con todo lo registrado.		
ID	Iteración		
H4	Sprint 4		
ID	Tarea	Voluntario	Total, de Hrs.
T1	Definir las reglas de negocio para el correcto funcionamiento.	Raul Guerrero Ramírez	22Hrs.
T2	Diseño para hoja o membrete para reportes.	Raul Guerrero Ramírez	22Hrs.
T3	Maquetación del módulo de finanzas.	Raul Guerrero Ramírez	22Hrs.
T4	Prototipado de presupuesto e instalación del sistema final.	Raul Guerrero Ramírez	22Hrs.
T5	Programación de front-end y back-end del módulo “Finanzas”.	Antonio Juarez Andrade	22Hrs.
T6	Reporte finanzas.	Antonio Juarez Andrade	22Hrs.
T7	Levantamiento del servidor.	Antonio Juarez Andrade	22Hrs.

		Horas repartidas entre los días de desarrollo del sprint del 1 al 20																			
ID		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
T1		8	8	6																	
T2				8	8	6															
T3					8	8	6														
T4							8	8	6												
T5									8	8	6										
T6											8	8	6								
T7																		8	8	6	

TABLERO DE TAREAS INICIO DEL SP4 H4.

Historia de usuario	Tareas	En proceso	Concluido
H4	SP4 H4 T1	T1	
Departamento de Finanzas			
H4	SP4 H4 T2	T2	
Departamento de Finanzas			
H4	SP4 H4 T3	T3	
Departamento de Finanzas			
H4	SP4 H4 T4	T4	
Departamento de Finanzas			
H4	SP4 H4 T5	T5	
Departamento de Finanzas			
H4	SP4 H4 T6	T6	
Departamento de Finanzas			
H4	SP4 H4 T7	T7	
Departamento de Finanzas			

Porcentaje de avance				
0-20%	21-40%	41-60%	61-80%	100%
T1,T2,T3,T4,T5,T6,T7				

PRINCIPIO DE LEAN SP4 H4

El desarrollo de este sprint esta estrictamente sujeto a lo requerido en la historia de usuario, lo cual es crear un módulo dentro del sistema que permita asignar un presupuesto a un servicio donde permita agregar ingresos, egresos y obtener ganancias reportándolo en un PDF. Mediante este principio se evita la creación de funciones innecesaria que el usuario nunca usara y se agiliza los tiempos de desarrollo.

SP4 H4 T5 PROGRAMACIÓN DE FRONT-END Y BACK-END

FINANZAS.

El controlador destinado para este módulo es “FinanceController” y de igual manera se agregaron los métodos de agregar y eliminar como en los otros módulos, estos métodos son fundamentales para cada uno de los módulos ya que son las operaciones habituales de un registro en este caso no se agregó el método de editar porque no es necesario.

```

20 public function view_finance(){
21
22     $servicios=DB::table("servicios")->select("*")->get();
23     $finanzas=DB::table("finanzas")->select("*")->get();
24     return view('Finance.finance',compact("servicios","finanzas"));
25 }
26

```

Figura 4.104 Método de PHP situado en el controlador “FinanceController”, retorna la vista del módulo “Finanzas”.

Este método retorna la vista del módulo “Finanzas”, se hace una consulta a la base de datos en específico a la tabla se “Finanzas” donde están todos los registros existentes

junto con los registros de servicios ya que el módulo “Finanzas” es codependiente del módulo “Servicios”, se envían a la vista mediante el “compact”.

```

155 <div class="card">
156   <div class="card-body">
157     <div style="margin-bottom: 20px;">
158       <button class="btn" style="background-color: #F5B32B" data-toggle="modal" data-target="#agregar" >Agregar</button>
159     </div>
160     <div class="table-responsive">
161
162       <table class="table" style="width: 100%;">
163         <thead class="thead-dark">
164           <tr>
165             <th style="text-align: center;">Folio</th>
166             <th style="text-align: center;">Cliente</th>
167             <th style="text-align: center;">Ingresos</th>
168             <th style="text-align: center;">Egresos</th>
169             <th style="text-align: center;">Ganancias</th>
170
171           </tr>
172         </thead>
173         <tbody>
174           @foreach($services as $service)
175           @if($service->finanza=="si")
176             <tr class="marca" onclick="pasar_id('{{ $service->id }}')">
177               <td style="text-align: center;">
178                 CMZ-{{$service->id}}
179               </td>
180               <td style="text-align: center;">
181                 {{$service->nombre}}
182               </td>
183               <td style="text-align: center;">@@
184               </td>
185               <td style="text-align: center;">@@
186               </td>
187               <td style="text-align: center;">@@
188             </td>
189           @endif
190           @endforeach
191         </tbody>
192
193       </table>
194     </div>
195   </div>
196
197 </div>
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236

```

Figura 4.105 Estructura HTML de la Vista para el módulo “Finanzas”.

Esta es la vista que retorna el controlador, es una tabla donde se muestran más a detalle los datos de cada uno de los registros de las finanzas, y así poder eliminar, editar o agregar un nuevo registro, para llenar la tabla se usó un “foreach” que permite recorrer el arreglo donde se encuentran los datos de las finanzas que se envió mediante el método del controlador.



Figura 4.106 Vista final del módulo “Finanzas”.

Para el filtro de la tabla y el paginado se usó nuevamente la API de JavaScript “Datatable” que se agregó en el módulo de “Usuarios”.

Para agregar un nuevo registro de finanzas se creó un formulario simple mediante un modal de la misma manera que se usó para el módulo “Usuarios”, los modales de editar y agregar serán muy similares solo que se cambian algunas etiquetas para que el usuario final identifique que es lo que está haciendo si agrega, eliminar o edita.

```
<!-- modal de agregar presupuesto-->
<div class="modal fade" id="agregar" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-xl modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Agregar Presupuesto</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <svg xmlns="http://www.w3.org/2000/svg" width="26" height="26" fill="currentColor" class="bi bi-x-octagon" viewBox="0 0 16 16" >
            <path d="M4.54.54a.5.5 0 0 1 .893.06.214a.5.5 0 0 1 .353.14614.394 4.394a.5.5 0 0 1 .146.353v.214a.5.5 0 0 1 -.146.353l4.54.146zM5.1 1 5.1v5.8L5.1 15h5.814.1-4.1V5.1l10.9 1H5.1z"/>
          </svg>
        </button>
      </div>
      <form action="{{ url('dar_alta_finanza') }}" method="POST">
        @csrf
        <div class="modal-body">
          <div class="row">
            <div class="col-md-6" style="margin-bottom: 10px;">
              <label>Folio</label>
              <select class="js-example-basic-single form-control" style="width: 100%; height: 15px;" onchange="buscar_servicio(this.value); activar_envio(); " name="folio" id="folio">
                <option selected disabled>.:Selecctiona:</option>
                @foreach($services as $service)
                  @if($service->finanza=="no")
                    <option value="{{$service->id}}>CMZ-{{$service->id}}</option>
                  @endif
                @endforeach
              </select>
            </div>
            <div class="col-md-6" style="margin-bottom: 10px;">
              <label>Cliente / Empresa</label>
              <input type="text" name="cliente" id="cliente" class="form-control" readonly onchange="activar_envio(); ">
            </div>
            <div class="col-md-12">
              <label>Observación general del servicio</label>
              <textarea class="form-control" name="observaciones_g" id="obs" rows="10" readonly onchange="activar_envio(); "></textarea>
            </div>
          </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-danger" data-dismiss="modal">Cancelar</button>
          <button class="btn" disabled id="envio_button">Aceptar</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Figura 4.107 Estructura HTML del modal para agregar una nueva finanza a un servicio.

Este es el formulario que se encuentra dentro del modal de agregar nuevo presupuesto se activa mediante el botón de agregar como en todos los módulos con el fin que el usuario se familiarice con el diseño, solo son campos que el usuario debe de llenar, el “select” de folio se llena con los folios de los servicios, porque para agregar un presupuesto a un servicio debe de existir primero el servicio, para esto se agregara un servicio de ejemplo.

```

532     function buscar_servicio(id_servicio_select){
533
534     $.ajax({
535         url: "{{url('/get_servicios')}}"+'/'+id_servicio_select,
536         dataType: "json",
537         //context: document.body
538     }).done(function(service_data) {
539
540         if (service_data==null){
541             document.getElementById("cliente").value=null;
542             document.getElementById("obs").innerHTML="";
543         }else{
544             document.getElementById("cliente").value=service_data.nombre;
545             document.getElementById("obs").innerHTML=service_data.observaciones;
546
547         }
548     });

```

Figura 4.108 Función de JavaScript que llena los campos dependiendo la opción seleccionada,

Esta es una función de JavaScript que acompaña al modal de agregar presupuesto, su función es buscar los datos del servicio seleccionado y llenar los campos con dichos datos esto lo hace mediante Ajax, por medio del controlador “SearchesController” que es el encargado de hacer búsquedas rápidas a la base de datos.

La otra función que lo acompañara es para validar que los campos estén llenos, esto con el fin de que no se guarden campos vacíos, que, aunque para dar agregar un presupuesto a un servicio solo es activar un campo de la tabla “servicios”.

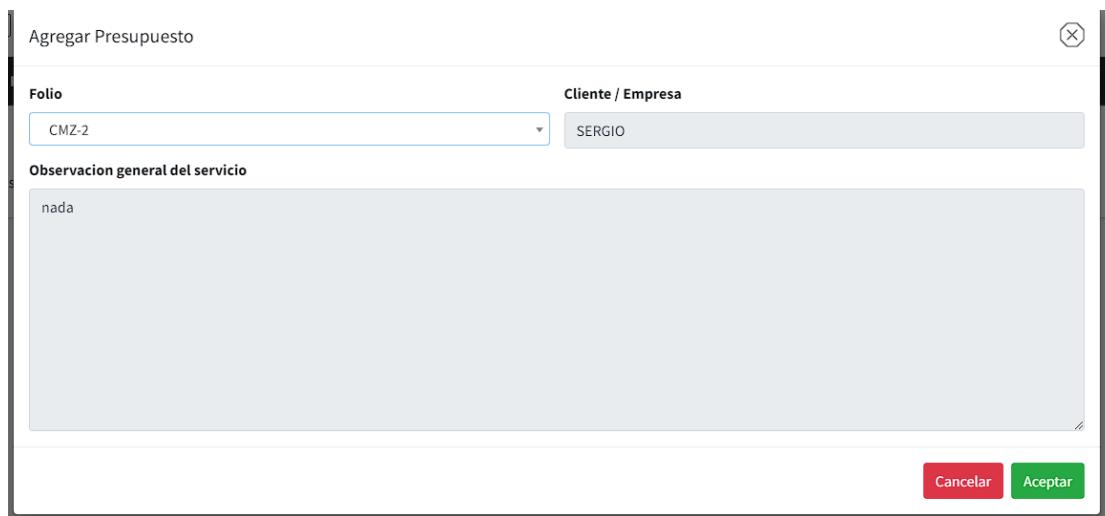


Figura 4.109 Vista del modal de agregar un nuevo Presupuesto a un servicio, situado en el módulo “Finanzas”.

Esta es la vista del formulario que será enviada al controlador para agregar el presupuesto al servicio seleccionado.

```
public function dar_alta_finanza(Request $request){
    try {
        DB::table("servicios")->where("id",$request["folio"])->update([
            "finanza"=>"si",
        ]);

        $service_data=DB::table("servicios")->where("id",$request["folio"])->first();

        $data =["usuario" => Auth::user()->name,"empresa" => $service_data->nombre,"id_servicio" => $service_data->id,"tipo" => "nuevo","total" => "0","cantidad" => "0"];
        Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Presupuesto Creado",$data,"finance"));

        return redirect()->back()->with(['message' => "Se creo la finanza correctamente", 'color' => 'success']);
        //echo "seguardo";
    } catch (\Exception $e) {
        return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
    }
}
```

Figura 4.110 Método de PHP situado en el controlador “FinanceController”, agrega un presupuesto a un servicio.

Este es el método del controlador que activa el presupuesto al servicio, solo es activar un campo del registro del servicio que indica que ya tiene presupuesto activo de igual manera enviamos un correo de alerta para que el administrador este consiente de lo que pasa en el sistema, cuando termina regresa a la vista anterior con un mensaje de éxito o de error según sea el caso.



Figura 4.111 Vista del correo electrónico que se envía al crear un nuevo presupuesto en un servicio.

Este es el correo que envía el sistema, en este caso le dice al usuario que fue en el módulo presupuesto o “Finanzas”, este formato será similar para los demás movimientos dentro de este módulo.

Figura 4.112 Vista del mensaje de éxito al crear un presupuesto a un servicio en el módulo “Finanzas”.

Esto indica que se activó el presupuesto al servicio, ya es visible en la tabla el presupuesto activo, pero sin datos porque todavía no se agregan ingresos ni egresos.

```
<!--menu de opciones de La tabla-->
<div id="menu_opciones" class="visible_off " style=" padding: 20px; background-color: #858585bd;">
    <button type="button" class="close" style="margin-right: -17px; margin-top: -20px; " onclick="cerrar_menu();">
        <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor" class="bi bi-x-octagon" viewBox="0 0 16 16" >
            </svg>
    </button>
    <button class="btn btn-danger" style="margin-bottom: 10px; font-weight: bold; margin-top: 10px; width: 100%;" data-toggle="modal" data-target="#eliminar_registro" onclick="eliminar_registro();">
        <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor" class="bi bi-pencil-square" viewBox="0 0 16 16" >
            </svg>
        Eliminar
    </button>
    <br>
    <button class="btn btn-warning" style="margin-bottom: 10px; font-weight: bold; width: 100%;" data-toggle="modal" data-target="#agregar_cantidad" onclick="ingreso();">
        <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor" class="bi bi-piggy-bank-fill" viewBox="0 0 16 16" >
            </svg>
        Registrar Ingreso
    </button>
    <br>
    <button class="btn btn-primary" style="margin-bottom: 10px; font-weight: bold; width: 100%;" data-toggle="modal" data-target="#agregar_cantidad" onclick="egreso();">
        <svg xmlns="http://www.w3.org/2000/svg" width="22" height="22" fill="currentColor" class="bi bi-cash-coin" viewBox="0 0 16 16" >
            </svg>
        Registrar Egreso
    </button>
    <br>
    <button class="btn btn-info" style="margin-bottom: 10px; font-weight: bold; width: 100%;" data-toggle="modal" data-target="#pdf" onclick="agregar_url();">
        <img alt="Icono de PDF" style="vertical-align: middle; margin-right: 5px;"/> PDF
    </button>
</div>
```

Figura 4.113 Estructura del menú despegable de la tabla para el módulo “Finanzas”.

Al menú despegable de la tabla se le agregaron más opciones que son las que serán la ocupadas para este módulo, no fue agregada la opción de editar por que para este módulo no es necesario, solo se agregan ingresos y egresos como método de edición.

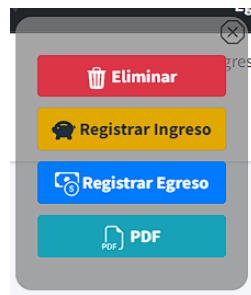


Figura 4.114 Vista del menú despegable de la tabla para el módulo “Finanzas”.

También tenemos PDF que mostrara todos los ingresos y egresos del sistema.

```
<!-- modal de agregar cantidad, este sera usado para ingreso y egreso-->
<div class="modal fade" id="agregar_cantidad" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-xl modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="texto_cantidad">Registrar Ingreso</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close" onclick="borrar_todo()"><span>×</span>
      </div>
      <form action="{{ url('/Agregar_cantidad') }}" method="POST">
        @csrf
        <div class="modal-body">
          <div class="row">
            <div class="col-md-3" style="margin-bottom: 10px;">
              <label>Concepto</label>
              <input type="text" name="concepto[]" id="concepto[]" class="form-control" onchange="activar_envio_2(); onkeyup="activar_envio_2();">
            </div>
            <div class="col-md-3" style="margin-bottom: 10px;">
              <label>Cantidad</label>
              <input type="number" name="cantidad[]" id="cantidad[]" class="form-control" onchange="activar_envio_2(); onkeyup="activar_envio_2();">
            </div>
            <div class="col-md-3" style="margin-bottom: 10px;">
              <label>Observaciones</label>
              <textarea class="form-control" name="observaciones[]" id="observaciones[]" rows="1" onchange="activar_envio_2(); onkeyup="activar_envio_2();"></textarea>
            </div>
            <div class="col-md-3" style="margin-bottom: 10px;">
              <br>
              <button type="button" class="btn btn-success" style="width:100%; margin-top: 8px;" onclick="agregar_registro()>Agregar</button>
            </div>
          </div>
          <div id="contenedor_padre">
            </div>
          </div>
        </div>
        <div class="modal-footer">
          <input type="hidden" name="tipo" id="tipo">
          <input type="hidden" name="id_servicio" id="id_ser">
          <button type="button" class="btn btn-danger" data-dismiss="modal" onclick="borrar_todo()>Cancelar</button>
          <button class="btn " disabled id="envio_button_2">Aceptar</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Figura 4.115 Estructura HTML del modal para agregar un ingreso o egreso a un servicio.

Este modal será para agregar ingresos y egresos al presupuesto, ya que las dos funciones cuentan con los mismos campos, solo serán modificados los títulos dependiendo si es ingreso o egreso.

```
551   function ingreso(){
552     document.getElementById('texto_cantidad').innerHTML='Registrar Ingreso';
553     document.getElementById('tipo').value='ingreso';
554     document.getElementById("id_ser").value=id_servicio;
555   }
556 }
```

Figura 4.116 Función de JavaScript que indica que la operación que se hará será un ingreso.

Esta es una de las funciones de JavaScript que acompañan al modal, lo que se encarga esta función es cambiar los títulos a “ingreso”, esto con el fin de que el usuario sepa que lo que está agregando es un ingreso.

```

557     function egreso(){
558         document.getElementById('texto_cantidad').innerHTML='Registrar Egreso';
559         document.getElementById('tipo').value='egreso';
560         document.getElementById("id_ser").value=id_servicio;
561     }

```

Figura 4.117 Función de JavaScript que indica que la operación que se hará será un egreso.

Esta función de JavaScript se encarga de cambiar los títulos por “egreso” con el fin de que el usuario entienda que está agregando un egreso, todo esto se hizo porque se ocupó el mismo modal para las dos acciones.

```

var j=1;
function agregar_registro(){

    if(j>0){
        $("#contenedor_padre").append(
            '<div id="contenedor_hijo'+j+'>' +
            '  <div class="row">' +
            '    <div class="col-md-3" style="margin-bottom: 10px;">' +
            '      <label>Concepto</label>' +
            '      <input type="text" name="concepto[]" id="concepto[]" class="form-control" onchange="activar_envio_2();"' +
            '          onkeyup="activar_envio_2();"' +
            '    </div>' +
            '    <div class="col-md-3" style="margin-bottom: 10px;">' +
            '      <label>Cantidad</label>' +
            '      <input type="number" name="cantidad[]" id="cantidad[]" class="form-control" onchange="activar_envio_2();"' +
            '          onkeyup="activar_envio_2();"' +
            '    </div>' +
            '    <div class="col-md-3" style="margin-bottom: 10px;">' +
            '      <label>Observaciones</label>' +
            '      <textarea class="form-control" name="observaciones[]" id="observaciones[]" rows="1" onchange="activar_envio_2();"' +
            '          onkeyup="activar_envio_2();"' +
            '    </div>' +
            '    <div class="col-md-3" style="margin-bottom: 10px;">' +
            '      <br>' +
            '      <button type="button" class="btn btn-danger eliminar_hijo" style="width:100%; margin-top: 8px;" id="'+j+''
            '          >Eliminar</button>' +
            '    </div>' +
            '  </div>' +
            '</div>';
        );
        j++;
    }
    //console.log(contador);
    activar_envio_2();
}

```

Figura 4.118 Función de JavaScript que permite agregar campos dinámicos al modal de agregar ingreso o egreso.

Esta función al igual que en el módulo de “Servicios” agrega código HTML a la página, agrega más campos si así lo desea el usuario, con el fin de agregar más de un ingreso o egreso según sea el caso, se activa al presionar el botón de “agregar” que está dentro del modal.

```

597     $(document).on('click', '.eliminar_hijo', function(){
598         var id=$('#this').attr("id");
599         $('#contenedor_hijo'+id+'').remove();
600         //j--;
601         activar_envio_2();
602     });
603

```

Figura 4.119 Función de JavaScript que elimina los campos dinámicos en el modal de agregar ingreso o egreso.

Esta función lo que hace es eliminar el código HTML agregado dinámicamente por el botón “agregar”, con el fin que el usuario elimine los campos que no le sirvan, se activa con el botón “eliminar” que contiene los nuevos campos creados. El llamado de la función “activar_envio_2()” valida si todos los campos fueron llenados, esta función está presente en todos los módulos, pero con distintos nombres.

Concepto	Cantidad	Observaciones	
cambio de llantas	2000	ninguna	Agregar
Concepto	Cantidad	Observaciones	
pintura	500	ninguna	Eliminar

Figura 4.120 Vista del modal de agregar ingreso en el módulo de “Finanzas”.

Este el modal de agregar ingreso están presentes los botones que agregan y eliminan los campos dinámicos que el usuario agregue, no tiene límite de campos.

```

public function agregar_presupuesto(Request $request){
    $contador=0;
    $total=0;
    try {
        foreach ($request["concepto"] as $clave => $dato) {
            DB::table("finanzas")->insert([
                "id_servicio"=>$request["id_servicio"],
                "concepto"=>$request["concepto"][$clave],
                "cantidad"=>$request["cantidad"][$clave],
                "tipo"=>$request["tipo"],
                "observaciones"=>$request["observaciones"][$clave],
            ]);
            $contador++;
            $total+=$request["cantidad"][$clave];
        }
        $service_data=DB::table("servicios")->where("id",$request["id_servicio"])->first();
        if ($request["tipo"]=="ingreso") {
            $data =[ "usuario" => Auth::user()->name, "empresa" => $service_data->nombre, "id_servicio" => $service_data->id, "tipo" => "ingreso", "total" => $total, "cantidad" => $contador];
            Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Ingreso Registrado",$data,"finance"));
            return redirect()->back()->with(['message' => "Se agrego el ingreso correctamente", 'color' => 'success']);
        }else{
            $data =[ "usuario" => Auth::user()->name, "empresa" => $service_data->nombre, "id_servicio" => $service_data->id, "tipo" => "egreso", "total" => $total, "cantidad" => $contador];
            Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Egreso Registrado",$data,"finance"));
            return redirect()->back()->with(['message' => "Se agrego el egreso correctamente", 'color' => 'success']);
        }
    } catch (\Exception $e) {
        return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
    }
}

```

Figura 4.121 Método de PHP situado en el controlador “FinanceController”, agrega un ingreso a un servicio.

Este es el método encargado de agregar ingresos y egresos, de igual manera que el modal este método hace las dos funciones, agrega los registros a una tabla llamada “finanzas”, una vez que se agregan los registros se envía un correo de alerta al administrador para que este consiente de lo que está pasando en el sistema, regresa a la vista anterior con un mensaje de éxito o de error según sea el caso.



Figura 4.122 Vista del mensaje de éxito al agregar un ingreso a un servicio en el módulo “Finanzas”.

Ya podemos ver las sumas de los ingresos en la tabla, falta agregar egresos, como ya se mencionó se ocupó el mismo modal y el mismo método del controlador para hacer las dos acciones.

Concepto	Cantidad	Observaciones
choque	8000	el chofer no estaba consiente.
Concepto	Cantidad	Observaciones
liquidación	20000	se despidió al chofer.

Agregar **Eliminar**

Cancelar **Aceptar**

Figura 4.123 Vista del modal de agregar egreso en el módulo de “Finanzas”.

Son datos a manera de ejemplo, el modal y las funciones de JavaScript son las mismas puesto que es el mismo modal y el mismo método del controlador.

Folio	Cliente	Ingresos	Egresos	Ganancias
CMZ-2	SERGIO	2500	28000	hay perdidas 25500

Mostrando registros del 1 al 1 de un total de 1 registros

Anterior **1** Siguiente

Figura 4.124 Vista del mensaje de éxito al agregar un egreso a un servicio en el módulo “Finanzas”.

Ya tenemos las dos, ingresos y egresos, se tiene más en egresos se obtiene perdidas.

```
<!-- modal de eliminar presupuesto-->
<div class="modal fade" id="eliminar_registro" tabindex="-1" aria-labelledby="exampleModallLabel" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModallLabel">Eliminar Presupuesto</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <svg xmlns="http://www.w3.org/2000/svg" width="26" height="26" fill="currentColor" class="bi bi-x-octagon" viewBox="0 0 16 16" >
            <path d="M4.54.146A.5.5 0 0 1 4.893 .0h6.214a.5.5 0 0 1 .353.14614.394 4.394a.5.5 0 0 1 .146.353v6.214a.5.5 0 0 1 -.146.353L4.54.146zM5.1 1 5.1v5.8L5.1 15h5.814.1-4.1V10.9 1H5.1z"/>
            <path d="M4.646 4.646a.5.5 0 0 1 .708.0L8 7.293l2.646-.2647a.5.5 0 0 1 .708.708L8.707 812.647 2.646a.5.5 0 0 1 -.708.708L8 8.7071-2.646 2.647a.5.5 0 0 1 -.708-.708L7.293 8 4.646 5.354a.5.5 0 0 1 0-.708z"/>
          </svg>
        </button>
      </div>
      <form action="{{ url('/Eliminar_finance') }}" method="POST">
        @csrf
        @method('DELETE')
        <div class="modal-body">
          <label style="font-weight: bold; font-size: 25px;">¿Seguro que quieres eliminar el Presupuesto?</label><br><br>
          <div style="text-align: center;">
            <label style="font-weight: bold; font-size: 25px;">Folio:</label>
            <label style="color: red; font-weight: bold; font-size: 25px;" id="text_folio"></label><br>
            <label style="font-weight: bold; font-size: 25px;">Cliente:</label>
            <label style="color: red; font-weight: bold; font-size: 25px;" id="text_cliente"></label>
          </div>
        </div>
        <div class="modal-footer">
          <input type="hidden" name="id_servicio_edit" id="id_servicio_edit_2">
          <button type="button" class="btn btn-secondary" data-dismiss="modal">Cancelar</button>
          <button class="btn btn-danger">Eliminar</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Figura 4.125 Estructura HTML del modal de eliminar el presupuesto de un servicio.

Este es el formulario de eliminar el registro, es igual que los demás modales de los otros módulos, solo es una alerta para que el usuario este seguro de eliminarlo.

```
610     function eliminar_registro(){
611
612     $.ajax({
613       url: "{{url('/get_servicios')}}"+'/' +id_servicio,
614       dataType: "json",
615       //context: document.body
616     }).done(function(service_data) {
617       if(service_data==null){
618         document.getElementById("text_folio").innerHTML=null;
619         document.getElementById("id_servicio_edit_2").value=null;
620         document.getElementById("text_cliente").innerHTML=null;
621       }else{
622         document.getElementById("text_folio").innerHTML="CMZ-"+service_data.id;
623         document.getElementById("id_servicio_edit_2").value=service_data.id;
624         document.getElementById("text_cliente").innerHTML=service_data.nombre;
625       }
626
627     });
628
629   }
630 }
```

Figura 4.126 Función de JavaScript que indica que registro será eliminado de la DB.

Esta es la función de JavaScript que llena los campos con los datos del registro y el usuario este completamente seguro.

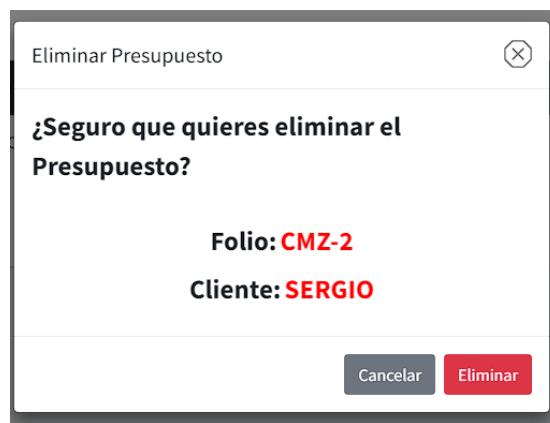


Figura 4.127 Vista del modal de eliminar presupuesto de un servicio en el módulo de “Finanzas”.

Esta es la alerta de eliminar, solo es un modal que le dice al usuario que será eliminado el presupuesto definitivamente de la base de datos.

```
public function eliminar_finance(Request $request){
    try {
        if ($request["id_servicio_edit"]!=0) {
            DB::table("finanzas")->where("id_servicio",$request["id_servicio_edit"])->delete();
            DB::table("servicios")->where("id",$request["id_servicio_edit"])->update([
                "finanza"=>"no",
            ]);
            $service_data=DB::table("servicios")->where("id",$request["id_servicio_edit"])->first();
            $data =[ "usuario" => Auth::user()->name,"empresa" => $service_data->nombre,"id_servicio" => $service_data->id,"tipo" => "eliminar",
                    "total" => "0","cantidad" => "0"];
            Mail::to("nuckelavee95@gmail.com")->send(new MessageSent("Presupuesto Eliminado",$data,"finance"));
        }
        return redirect()->back()->with(['message' => "Se Elimino correctamente el Presupuesto", 'color' => 'danger']);
    } catch (\Exception $e) {
        return redirect()->back()->with(['message' => "Algo salio mal con la base de datos, intente de nuevo", 'color' => 'warning']);
    }
}
```

Figura 4.128 Método de PHP situado en el controlador “FinanceController”, eliminar el presupuesto de un servicio.

Este es el método de eliminar un presupuesto, se eliminan los registros de los ingresos y egresos y al registro del servicio solo indicamos que ya no tiene presupuesto activo, una vez que termina se envía una alerta por correo al administrador. Se regresa a la vista anterior con un mensaje de éxito o de error según sea el caso.



Figura 4.129 Vista del mensaje de alerta al eliminar un presupuesto de un servicio en el módulo de “Finanzas”.

Las rutas del módulo “Finanzas” son las siguientes:

```

64 //finance
65 Route::get('/Finanzas',[FinanceController::class,'view_finance'])->name('finance');
66 Route::post('/dar_alta_finanza',[FinanceController::class,'dar_alta_finanza'])->name('dar_alta_finanza');
67 Route::post('/Aregar_cantidad',[FinanceController::class,'agregar_presupuesto'])->name('agregar_presupuesto');
68 Route::delete('/Eliminar_finance',[FinanceController::class,'eliminar_finance'])->name('eliminar_finance');
69 Route::get('/PDF_finance/{id}',[FinanceController::class,'pdf_finance'])->name('pdf_finance');
70

```

Figura 4.130 Rutas que activan los métodos del controlador “FinanceController”.

Cada ruta tiene su “url” con la cual el usuario final hará consulta y activara el método correspondiente de la clase “FinanceController” que es el controlador de este módulo.

SP4 H4 T6 REPORTE FINANZAS.

Para agregar el PDF de presupuesto se tiene que crear la vista donde se mostraran dichos datos, se usar DOMPDF como en el módulo de “Servicios” y de igual manera se usara el mismo modal que muestra el PDF de “Servicios”.

```

112
113     public function pdf_finance($id){
114
115         $datos=DB::table("servicios")->where("id",$id)->first();
116         $servicio_vehiculos=DB::table("servicio_vehiculo")->where("id_servicio",$id)->get();
117         $finanzas=DB::table("finanzas")->where("id_servicio",$id)->get();
118         $pdf = PDF::loadView("Finance.pdf.finance",compact("datos","servicio_vehiculos","finanzas"))->setPaper(array(0,0,1186,1536));
119         //$nombre_pdf="Matriz Master_".\$proyectos->nombre.".pdf";
120         return $pdf->stream("folio_CMZ-".$id.".pdf");
121     }
122

```

Figura 4.131 método de PHP situado en el controlador “FinanceController”, retorna el PDF para el módulo “Finanzas”.

Al igual que en el módulo de “Servicios” se hace consulta a la base de datos de los registros y se envían mediante “compact” a la vista donde se mostrarán los datos del presupuesto para el servicio.

```

<p style=" margin-top: 55px; padding-left: 50px; font-size: 30px;">Finanzas</p>

<table class="table" style="border-collapse:separate; width: 100%; border-spacing:0 10px;">
  <tbody>
    <tr style="font-weight: 0; ">
      <td style="font-size: 30px; text-align: center;" colspan="4">Ingresos</td>
    </tr>
    <tr style="font-weight: 0; "> ==
    </tr>
    <?php $ingresos=0 ?>
    @foreach($finanzas as $finanza)
    @if($finanza->tipoo=="ingreso")
    <tr style="font-weight: 0; "> ==
    </tr>
    <?php $ingresos+=$finanza->cantidad; ?>
    @endif
    @endforeach
  </tbody>
</table>

<table class="table" style="border-collapse:separate; width: 100%; border-spacing:0 10px;">
  <tbody>
    <tr style="font-weight: 0; ">
      <td style="font-size: 30px; text-align: center;" colspan="4">Egresos</td>
    </tr>
    <tr style="font-weight: 0; "> ==
    </tr>
    <?php $egresos=0 ?>
    @foreach($finanzas as $finanza)
    @if($finanza->tipoo=="egreso")
    <tr style="font-weight: 0; "> ==
    </tr>
    <?php $egresos+=$finanza->cantidad; ?>
    @endif
    @endforeach
  </tbody>
</table>

<table class="table" style="border-collapse:separate; width: 100%; border-spacing:0 55px;">
  <tbody>
    <tr style="font-weight: 0; ">
      <td style="font-size: 30px; text-align: center;" colspan="4">Totales</td>
    </tr>
    <tr style="font-weight: bold;"> ==
    </tr>
  </tbody>
</table>

<p style=" margin-top: 55px; padding-left: 50px; position: absolute; font-size: 30px;"><br>Observaciones generales del servicio: <br>{$datos->observaciones}</p>
```

Figura 4.132 Estructura HTML de la vista para el PDF del módulo “Finanzas”.

Esta es parte del PDF del módulo “Finanzas”, este se une con el PDF de “Servicios” en la parte superior se agregó el HTML de la vista del PDF de “Servicios”, en la parte inferior se agregó las sumas y restas de los ingresos y egresos de dicho servicio mediante el módulo “Finanzas”, el módulo de “Finanzas” depende totalmente del módulo “Servicios”. Se usaron tablas que permiten una facilidad al acomodar el texto y ayuda cuando existen muchos registros y el acomodo sea automático. Se recorre las variables enviadas mediante el “compact” ya que son arreglos y se tiene que recorrer mediante un “foreach”.

```
<!-- modal de pdf-->
<div class="modal fade" id="pdf" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog modal-xl modal-dialog-centered">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Ver PDF</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <svg xmlns="http://www.w3.org/2000/svg" width="26" height="26" fill="currentColor" class="bi bi-x-octagon" viewBox="0 0 16 16" >
            <path d="M4.54.146a.5.5 0 0 1 4.893 0h6.214a.5.5 0 0 1 .353.146l4.394 4.394a.5.5 0 0 1 .146.353v6.214a.5.5 0 0 1 -.146.353l-.146.353a.5.5 0 0 1 4.394a.5.5 0 0 1 -.353.146H4.893a.5.5 0 0 1 -.353.146L146.146 11.46a.5.5 0 0 1 11.107V4.893a.5.5 0 0 1 .146-.353L4.54.146zM5.1 1 5.1v5.8L5.1 15h5.8l4.1-4.1V5.11L0.9 1H5.1z"/>
          </svg>
        </button>
      </div>
      <div class="modal-body">
        <div class="col-md-12" style="text-align: center; display: none;" id="no_se_mira">
          <p>UPss! &nbsp;&nbsp; !CREO QUE NO SE VE BIEN EL PDF; VAMOS A OTRA PAGINA OK!</p>
          <a class="btn btn-success" target="_blank" href="" id="ir_otro_lado">¡VAMOS! <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-box-arrow-up-right" viewBox="0 0 16 16" >
            <path fill-rule="evenodd" d="M8.636 3.5a.5.5 0 0 0 0 1v16a.5.5 0 0 0 1 16h10a1.5 1.5 0 0 0 1.5 16h10a1.5 1.5 0 0 0 1.5-1.5V7.864a.5.5 0 0 0 0 1v14.5a.5.5 0 0 1 .5-1.5h-10a.5.5 0 0 0 1-.5-1.5V-10a.5.5 0 0 1 .5-.5h6.636a.5.5 0 0 0 .5-.5z"/>
          </path>
        </a>
        <embed type="application/pdf" src="" style="width:100%; height: 600px;" id="visor_pdf">
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-success" data-dismiss="modal">Aceptar</button>
      </div>
    </div>
  </div>
</div>
```

Figura 4.133 Estructura HTML del modal para visualizar el PDF del módulo “Finanzas”.

Este es el modal para visualizar el PDF, es exactamente el mismo que se usó para visualizar el PDF del módulo “Servicios”, de la misma forma va acompañado con la función de JavaScript que agrega el “id” a la “url”.

```
71     Route::get('/PDF_servicio/{id}', [ServicesController::class, 'pdf_service'])->name('pdf_service');
72
```

Figura 4.134 Ruta para visualizar el PDF para el módulo “Finanzas”.

Esta es la ruta que activa el método para mostrar el PDF, pertenece al controlador “ServicesController”, pide un parámetro que en este caso es el “id” del registro y así poder hacer un filtrado en la base de datos a base de ese “id”.

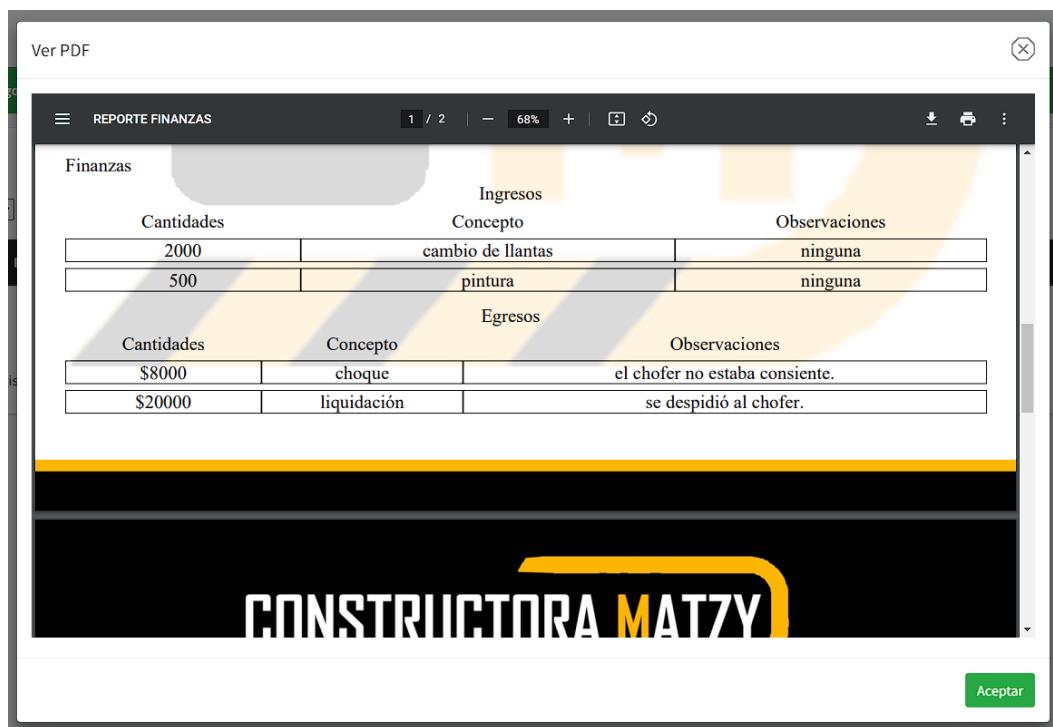


Figura 4.135 Vista de modal para visualizar el PDF en el módulo “Finanzas”.

Así es como se muestra el PDF del presupuesto, el reporte contiene todos los datos del servicio creado más los ingresos y egresos, permite descargarlo, imprimirla las funciones que encontramos al visualizar un PDF en el navegador. Esta forma es más útil que mandar al usuario a otra vista, muestra el PDF de una forma rápida para poder imprimirla o descargarla sin salir del sistema, de igual manera para poder generar los PDF debes de iniciar sesión, esto para evitar filtración de información y así se garantiza la integridad de los datos de la empresa.

SP4 H4 T7 Levantamiento del servidor.

Instalación

Se optó por instalar el sistema en una computadora que también funcionara como servidor mediante la apertura de puertos en el Router, esto permite que cualquier persona con acceso a internet pueda hacer uso del sistema mediante cualquier dispositivo.

Características del equipo de instalación.

- Modelo: HP All in One 18-155010la.
- Procesador: Celeron Trail-D J1800.
- Gráficos: Integrados.
- Almacenamiento: HDD 500 GB.
- Memoria: RAM 8 GB.
- Sistema operativo: Windows 10 Pro.



Figura 4.136 Equipo de cómputo que funcionara como servidor local.

Características de la conexión

- Proveedor: Telmex .
- Velocidad: 100-125 MB.
- Tipo de conexión: RJ45/CAT6/1000MB.

Características del Servidor

- Proveedor: NO-IP.
- Puerto SQL: 3306.
- Puerto HTTP: 80.

- Dominio: axolotlsw.sytes.net.
- Dirección web: http://axolotlsw.sytes.net/matzy_system/public/

Preparando el Router.

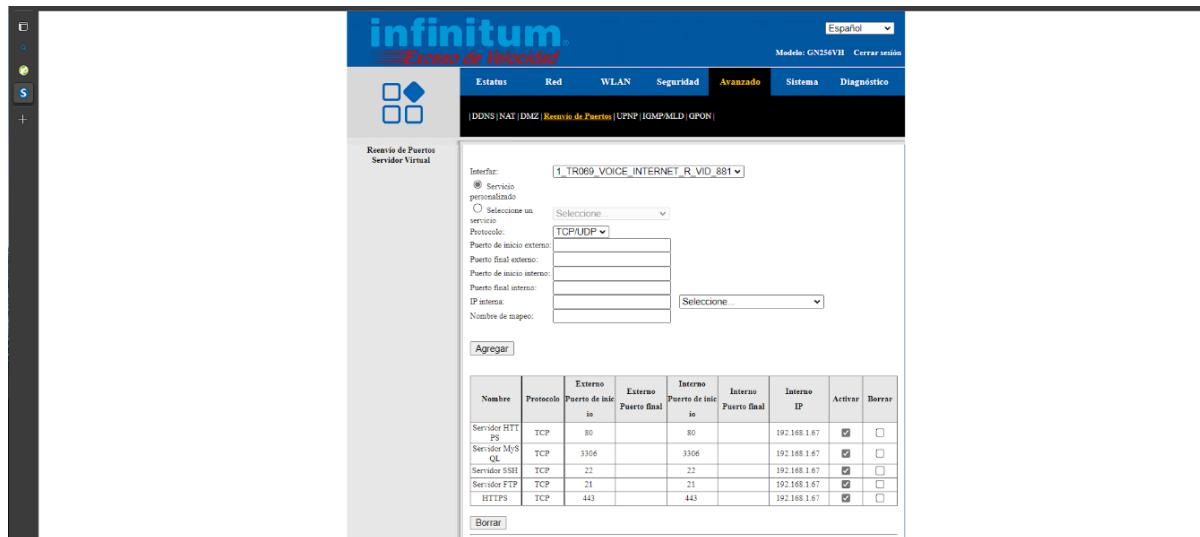


Figura 4.137 Activación de puestos en el Router para el servidor.

Se agregan los puestos de cada uno de los servicios que utilizará el servidor para su funcionamiento.

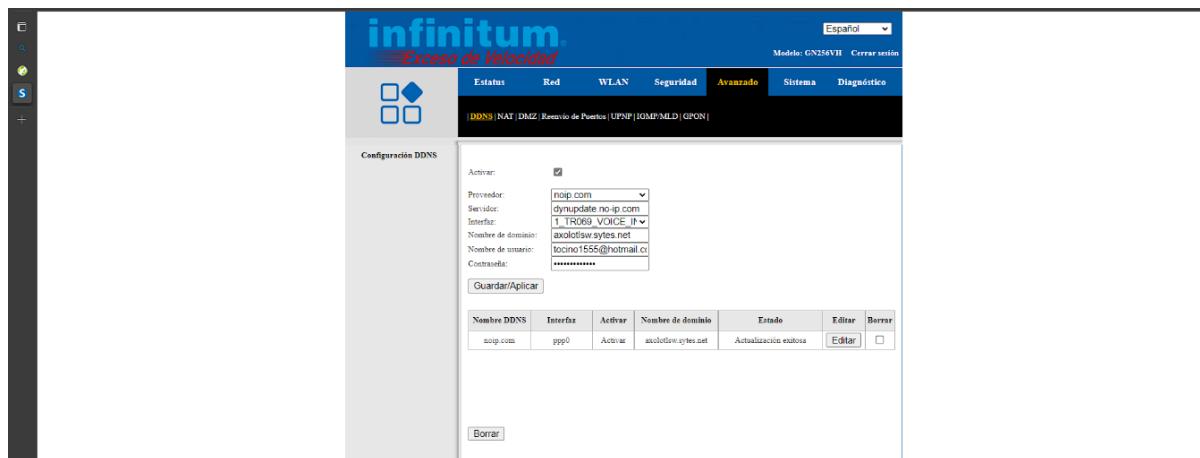


Figura 4.138 Apartado para agregar el dominio y el tipo de servicio que se usara.

Se indica cual será el proveedor del dominio, por defecto TELMEX ya tiene establecido el servicio de NO-IP.

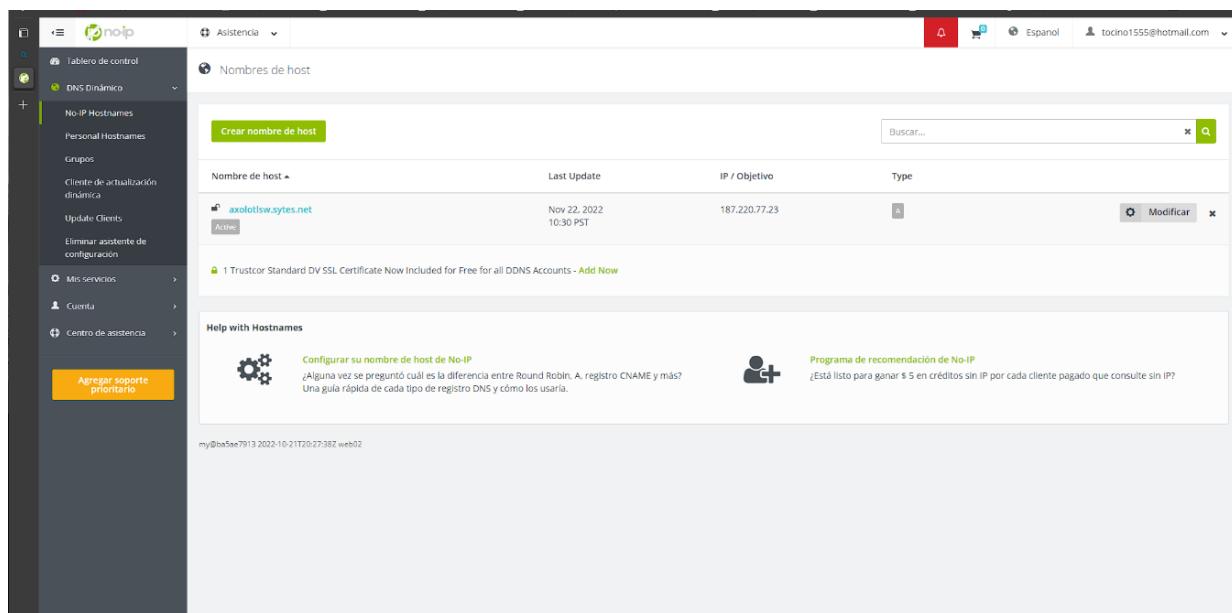


Figura 4.139 Creación del dominio por el servicio gratuito de NO-IP.

Creación del dominio para poder acceder al sistema a través de internet.

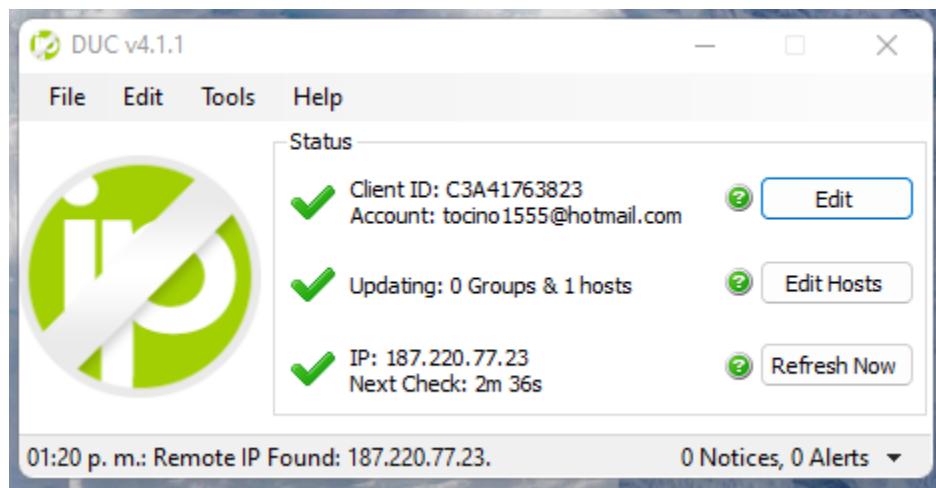


Figura 4.140 Activación del servidor local.

Servidor encendido y funcionando correctamente en la computadora.

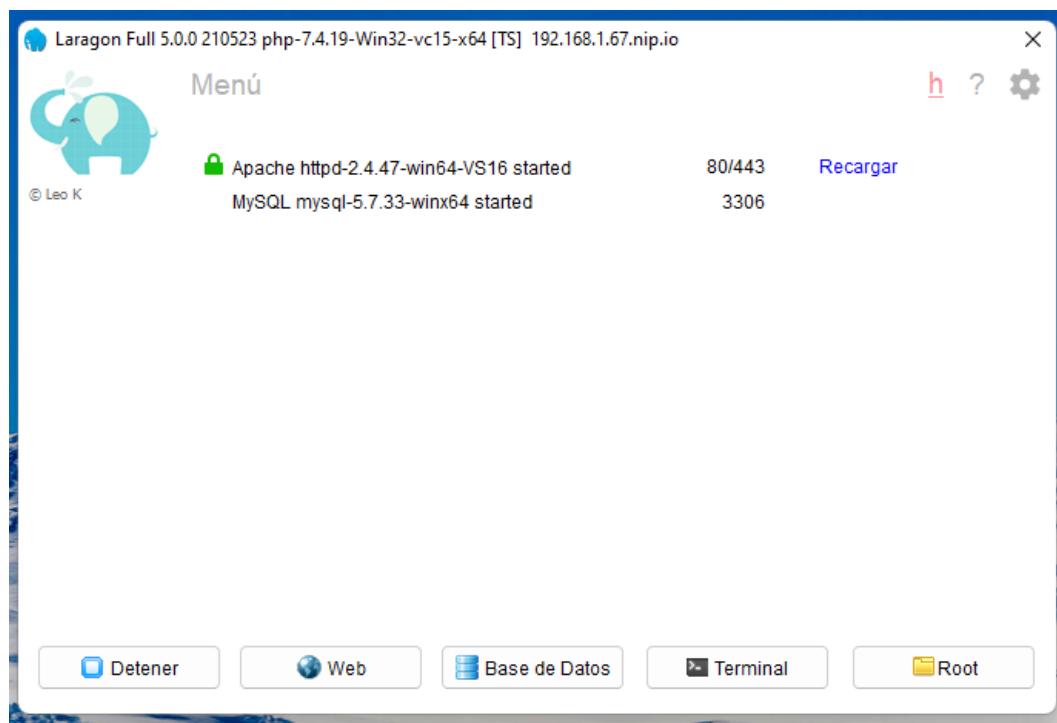


Figura 4.141 Activación de MySQL y Apache.

Se encienden los puestos de Apache y MySQL.

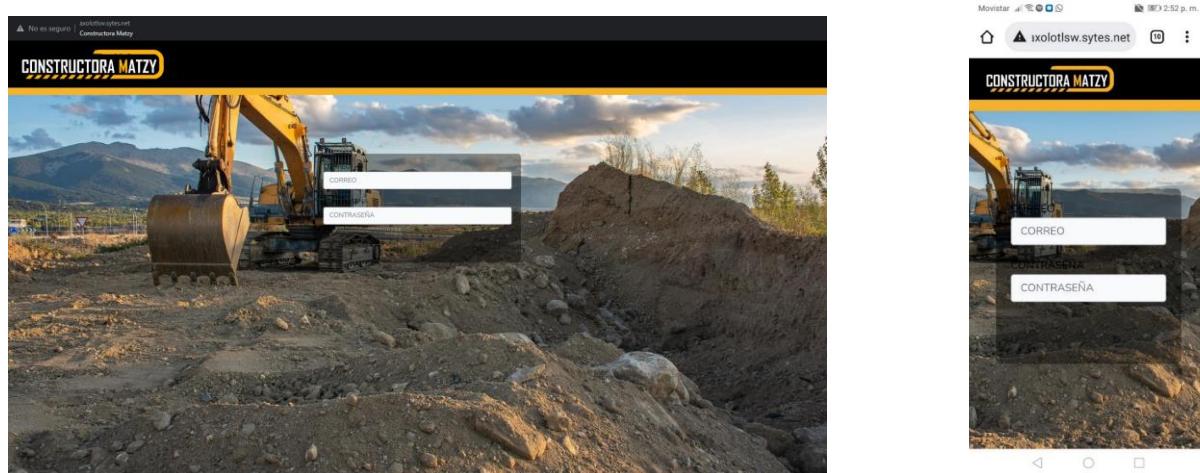


Figura 4.142 Resultados de la activación del servidor y sistema concluido, vistas de diferentes dispositivos.

Sistema ejecutándose perfectamente y está siendo consultado desde otros dispositivos.

SP4 H4 LIBERACIÓN Y ENTREGA DEL SPRINT DEPARTAMENTO DE FINANZAS.

Entrega de Sprint

Información de la empresa y proyecto:

Empresa/ Organización	Constructora Matzy S.A de C.V.
Proyecto	Matzy System

información del sprint:

Iteración/Sprint	ID	Enunciado de la historia
SP4 Departamento de finanzas	H4	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita asignar un presupuesto a un servicio que ya está iniciado, con la finalidad de que ahí pueda registrar mis ingresos y egresos correspondientes a este servicio con el cliente en turno y así obtener las ganancias en este apartado, obteniendo un reporte interno con todo lo registrado.

Información de la entrega:

Numero de iteración/ Sprint	SP4 H4/Departamento de finanzas
Persona responsable del recibimiento	
Cargo	Dueño de la empresa

información del equipo de trabajo:

Nombre	Rol
Raul Guerrero Ramírez	Scrum master
Antonio Juárez Andrade	Equipo de trabajo

Observaciones:

Bajo el principio de Lean, sujeto a la o las demandas del cliente o dueño de la empresa, el sprint fue entregado con cada uno de los requerimientos citados en el enunciado de la historia, dando por concluido al cien por ciento el total de las tareas impuestas para el sprint SP4 H4 Departamento de finanzas

Firma de conformidad del encargado
de recibir el sprint



✓ Representante legal.
"Constructora Matzy"

Figura 4.143 Hoja de liberación del sprint “departamento de finanzas”.

TABLERO DE TAREAS FINAL DEL SP4 H4.

Historia de usuario	Tareas	En proceso	Concluido
H4 Departamento de Finanzas	SP4 H4 T1		T1
H4 Departamento de Finanzas	SP4 H4 T2		T2
H4 Departamento de Finanzas	SP4 H4 T3		T3
H4 Departamento de Finanzas	SP4 H4 T4		T4
H4 Departamento de Finanzas	SP4 H4 T5		T5
H4 Departamento de Finanzas	SP4 H4 T6		T6
H4 Departamento de Finanzas	SP4 H4 T7		T7
Porcentaje de avance			
0-20%	21-40%	41-60%	61-80%
			100%
			T1,T2,T3,T4 T5,T6,T7

4.6 RESULTADOS OBTENIDOS

Los distintos módulos ya se encuentran concluidos y totalmente funcionales.

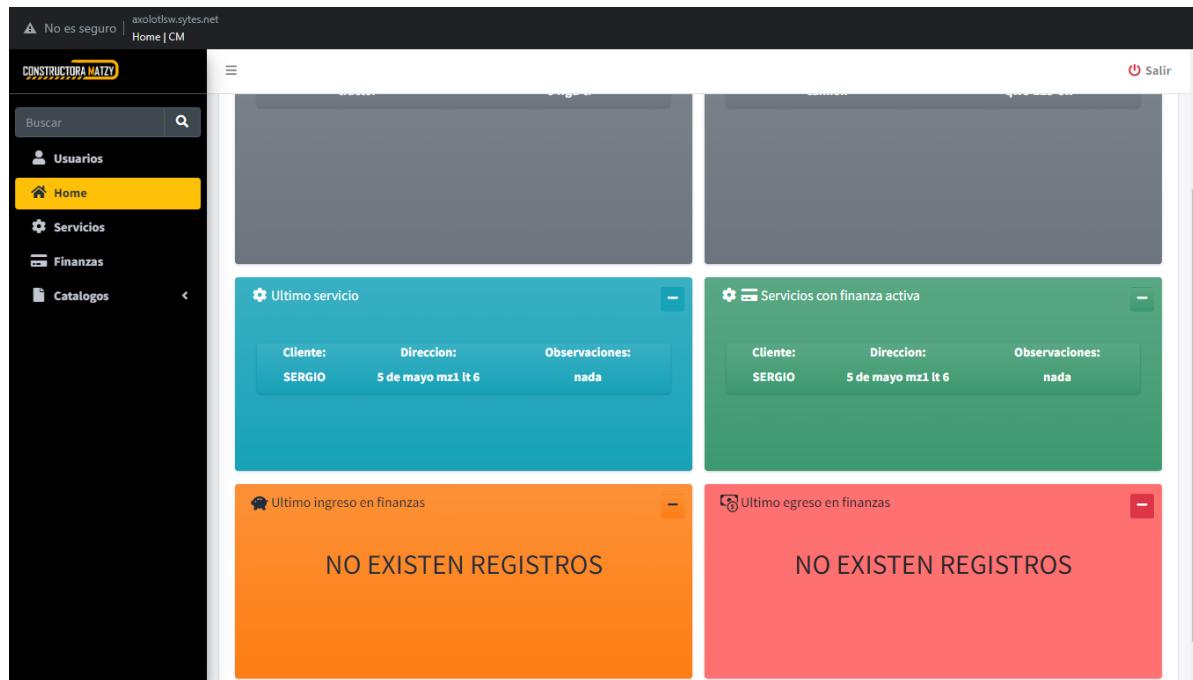


Figura 4.144 Vista final del módulo “Home” finalizado y vista final desde el servidor.

Modulo “HOME”.

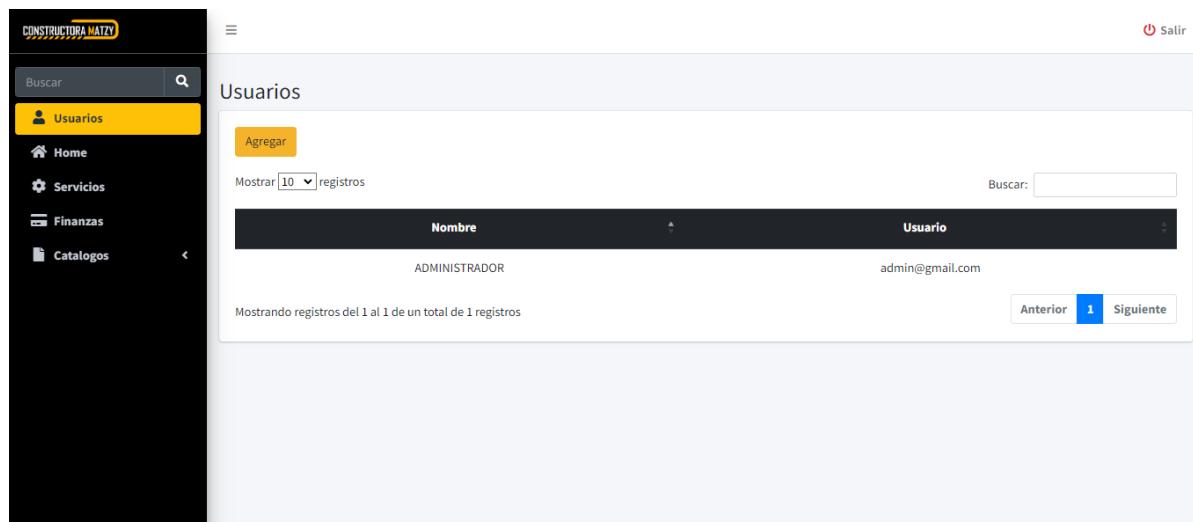


Figura 4.145 Vista final del módulo “Usuarios” finalizado y vista final desde el servidor.

Modulo “usuarios”, recordemos que es donde se agregan a nuevos usuarios al sistema.

Figura 4.146 Vista final del módulo “Servicios”.

Modulo “servicios”, recordemos que aquí es donde se crean, editan y eliminan los servicios, así como generar el reporte en PDF.

Figura 4.147 Vista final del módulo “Finanzas”.

Modulo “finanzas”, recordemos que aquí es donde se generan presupuestos a un servicio activo, generando un PDF con todos los datos del servicio más los ingresos, egresos y ganancias. El menú despegable que se muestra está presente en todos los módulos.

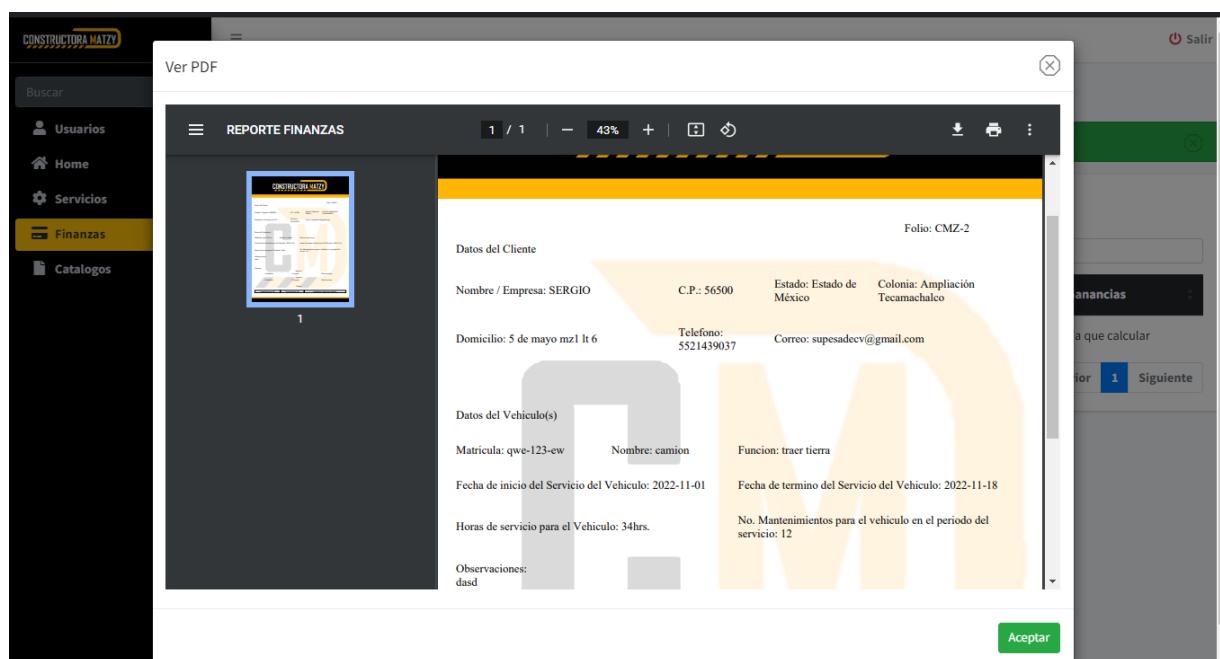


Figura 4.148 vista final del modal para el PDF del módulo “Finanzas”.

Así es como se genera el PDF en el sistema, esta vista está presente en servicios y en finanzas, los dos apartados generar reportes en PDF.

Matrícula	Nombre	Función
54fdg-tr	tractor	quitar tierra
qwe-123-ew	camion	traer tierra

Figura 4.149 Vista Final del módulo “Catálogos”.

Modulo “catálogos”, en este módulo es donde se registran nuevos vehículos en el sistema para poder generar un servicio con dichos vehículos.

DOCUMENTO DE ACEPTACIÓN DE LA ENTREGA TOTAL DEL SOFTWARE

Documento de entrega del proyecto “Matzy System”

Información de la empresa y proyecto:

Empresa/ Organización	Constructora Matzy S.A de C.V.
Proyecto	Matzy System

información de los entregados sprints:

Iteración/Sprint	ID	Enunciado de la historia
SP1 Perfiles	H1	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" de seguridad, con la finalidad de que solo yo pueda acceder a la información en el software y "modulo" que me permita tener datos resumidos del sistema en general, con la finalidad de que pueda ver dicha información al iniciar el software.
SP2 Personal	H2	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita darles acceso a nuevos usuarios, con la finalidad de que parte de mi personal pueda acceder a la información una vez yo le proporcione su propio usuario y contraseña.
SP3 Departamento de servicios	H3	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita hacer la toma de datos necesarios por parte del cliente y la disponibilidad de mis vehículos y fechas en las que estos estarán disponibles y de esta manera, con la finalidad de que pueda decidir correctamente cuando iniciar un servicio, obteniendo un reporte interno con todo lo registrado y "modulo" que me permita registrar mis vehículos, con la finalidad de que no tenga que registrarlos cada vez.
SP4 Departamento de finanzas	H4	Yo "representante legal de la Constructora Matzy ", necesito de un "modulo" que me permita asignar un presupuesto a un servicio que ya está iniciado, con la finalidad de que ahí pueda registrar mis ingresos y egresos correspondientes a este servicio con el cliente en turno y así obtener las ganancias en este aparato, obteniendo un reporte interno con todo lo registrado.

Figura 4.150 Hoja uno del documento de entrega del proyecto.

Información de la entrega del proyecto:

Nombre del proyecto	Matzy System
Persona responsable del recibimiento	[REDACTED]
Cargo	Dueño de la empresa

Requerimientos del proyecto:

Requerimientos	Aceptado	No aceptado
Funcionales	✓	
No funcionales	✓	
Seguridad	✓	
Portabilidad	✓	

Los requerimientos funcionales de un sistema, son aquellos que describen cualquier actividad que este deba realizar, en otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones. Los requerimientos no funcionales se refieren a las cualidades, restricciones y características del software, a diferencia de los funcionales, no determinan una funcionalidad del sistema a desarrollar. Los requerimientos no funcionales se caracterizan por ser, específicos, diseños, cuantificables, verificables. Los requerimientos de seguridad hacen alusión a las partes de seguridad del software como registros, encriptación de usuarios, restricción de las DB etc. Los requerimientos de portabilidad se refieren a que el software es ejecutable en cualquier dispositivo ya sea en el total de sus funciones o parcialmente según sea el acuerdo.

Firma de conformidad del encargado
de recibir el proyecto.

✓ Representante legal.
"Constructora Matzy"

Figura 4.151 Hoja dos del documento de entrega del proyecto.

CARTA DE AGRADECIMIENTO



Constructora
Matzy, S.A. de C.V.

Ixtapaluca, Estado de México, a 01 de Enero del 2023

Asunto: Carta de agradecimiento

Tecnológico de Estudios Superiores
Del Oriente del Estado de México
Juárez Andrade Antonio
Presente.

Por este medio expreso mi agradecimiento al alumno **Juárez Andrade Antonio** De la carrera de **Ingeniería en Sistemas Computacionales**, inscrito en el **Tecnológico de Estudios Superiores del Estado de México** por su participación en el desarrollo del sistema interno administrativo que lleva por nombre "Matzy System", software que ayuda a optimizar la elaboración de formatos internos de la empresa **Constructora Matzy S.A. de C.V.** ubicada en [REDACTED] Colonia [REDACTED] automatizando formatos tales como reportes de finanzas, reportes de servicios, control de vehículos y envíos a correo electrónico, donde dentro de su participación estuvo el desarrollo de diseño UX, diseño UI, levantamiento de requisitos y dirección de proyecto a través de la metodología ágil llamada scrum.

Figura 4.152 Hoja de agradecimiento.

4.7 CONCLUSIONES

La utilidad del software en general puede ayudar a tener una mejor administración de los servicios que brinda la empresa, este software está hecho a la medida del cliente y por ende el sistema cumple con todas las expectativas esperadas, ayudando así al cliente a realizar sus tareas que le tomaría realizar en horas y con el sistema se haría en un minuto. Este software no es apto para adaptación a otra empresa, pero la metodología Axolotl puede ser replicada para futuros proyectos. Ceñirse a la demanda y no a la oferta agiliza los tiempos de desarrollo ya que solo se desarrolla lo que pide el cliente y así se evita perder tiempo desarrollando funciones que no pide y esperar que le guste al cliente.

Para que exista un buen desarrollo de software se debe tener una buena gestión de proyectos, el scrum debe estar al tanto de las capacidades del equipo de desarrollo, el cliente puede solicitar algunas tareas fuera del alcance del desarrollador, esto implica que las entregas de los avances empiecen a salir de la fecha estimada y el cliente empiece a desconfiar totalmente de la funcionalidad del sistema. El scrum debe dar alternativas que, si están a disposición del desarrollador, negociar tiempos de entrega o simplemente no aceptar el proyecto por desconocimiento del equipo de desarrollo, aunque el programador debe estar en constante aprendizaje, hay cosas que simplemente nunca se han hecho y requieren de más tiempo de desarrollo.

Las interfaces del sistema deben estar a la altura del estándar de diseño actual, esto ayuda al usuario a navegar por el software sin necesidad de capacitación, el diseño debe ser realizado por un diseñador capacitado que conozca el estándar de interfaz actual, ya que a menudo sucede que esto se deja al programador que carece de estos conocimientos y el cliente puede rechazar el software por el simple hecho de que no es fácil de usar y no se ve bien, tener una buena presentación del software hoy en día es muy solicitado y si no tienes tales estándares de diseño, el software tiende a fallar.

FUENTES DE INFORMACIÓN

5.1 REFERENCIAS DIGITALES

Altube Vera, R. (2021) Qué Es Laravel: Características y Ventajas, OpenWebinars.net. Available at: <https://openwebinars.net/blog/que-es-laravel-caracteristicas-y-ventajas/> (Accessed: September 15, 2022).

García, I.J.B. (2020) Backend y frontend, ¿Qué es y cómo funcionan en la programación?, Conectividad y Soluciones de TI. Available at: <https://www.servnet.mx/blog/backend-y-frontend-partes-fundamentales-de-la-programacion-de-una-aplicacion-web> (Accessed: September 18, 2022).

mozilla, D. (2021) CSS - aprende sobre desarrollo web: MDN, Aprende sobre desarrollo web | MDN. Available at: <https://developer.mozilla.org/es/docs/Learn/CSS> (Accessed: September 25, 2022).

mozilla, D. (2021) HTML: Lenguaje de etiquetas de hipertexto, MDN. Available at: <https://developer.mozilla.org/es/docs/Web/HTML> (Accessed: October 5, 2022).

mozilla, D. (2021) JavaScript, MDN. Available at: <https://developer.mozilla.org/es/docs/Web/JavaScript> (Accessed: October 10, 2022).

mozilla, D. (2021) MVC - Glosario de mdn web docs: Definiciones de términos relacionados con la web: MDN, Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN. Available at: [https://developer.mozilla.org/es/docs/Glossary/MVC#:~:text=MVC%20\(Modelo%2DVista%2DControlador,de%20negocios%20y%20su%20visualizaci%C3%B3n.](https://developer.mozilla.org/es/docs/Glossary/MVC#:~:text=MVC%20(Modelo%2DVista%2DControlador,de%20negocios%20y%20su%20visualizaci%C3%B3n.) (Accessed: October 22, 2022).

Edix (2022) Framework: Qué Es, para qué sirve y algunos ejemplos, Edix España. Available at: <https://www.edix.com/es/instituto/framework/> (Accessed: October 27, 2022).

Wikipedia (2022) PHP, Wikipedia. Wikimedia Foundation. Available at: <https://es.wikipedia.org/wiki/PHP> (Accessed: October 27, 2022).

Jetstream, L. (2022) # introduction, Laravel Jetstream. Available at: <https://jetstream.laravel.com/2.x/introduction.html> (Accessed: November 10, 2022).

Rodríguez, A. and Pio, L. (2021) Bootstrap: ¿Qué es, para qué sirve y cómo instalarlo?, Rock Content - ES. Available at: <https://rockcontent.com/es/blog/bootstrap/> (Accessed: November 18, 2022).

Leokhoa. (2019) Documentation, Laragon. Available at: <https://laragon.org/docs/> (Accessed: November 20, 2022).

Zúñiga, F.G. (2022) ¿Qué es phpmyadmin y cómo usarlo?, Blog de arsys.es. Available at: <https://www.arsys.es/blog/phpmyadmin> (Accessed: November 23, 2022).

Torrealba, R. (2020) Adminlte, Styde.net. Available at: <https://styde.net/tag/adminlte/> (Accessed: November 27, 2022).

IBM (2020) ¿Qué es ajax?, IBM. Available at: <https://www.ibm.com/docs/es/rational-soft-arch/9.6.1?topic=page-asynchronous-javascript-xml-ajax-overview> (Accessed: November 28, 2022)

GLOSARIO

6.1 GLOSARIO TÉCNICO

HTML: Lenguaje de Marcas de Hipertexto, del inglés (HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript).

CSS: Las hojas de estilo en cascada (Cascading Style Sheets), lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado como HTML, que especifica cómo se presentan los documentos a los usuarios: cómo se diseñan, compaginan, etc.

JavaScript: Lenguaje de programación basada en prototipos, multiparadigma, de un solo hilo, dinámico, con soporte para programación orientada a objetos, imperativa y declarativa (por ejemplo, programación funcional).

PHP: Lenguaje de programación de uso general que se adapta especialmente al desarrollo web.

Front-end: Parte del desarrollo web que se dedica a la parte frontal de un sitio web, en pocas palabras del diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos.

Back-end: Capa de acceso a los datos, ya sea de un software o de un dispositivo en general, es la lógica tecnológica que hace que una página web funcione, lo que queda oculto a ojos del visitante.

Framework: Esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, plantilla que sirve como punto de partida para la organización y desarrollo de software agilizando los tiempos de trabajo.

Laravel: Laravel es un framework de back-end para desarrollar aplicaciones y servicios web con PHP 5, PHP 7 y PHP 8. Su filosofía es desarrollar código PHP de forma

elegante y simple, evitando el "código espagueti". Fue creado en 2011 y tiene una gran influencia de frameworks como Ruby on Rails, Sinatra y ASP.NET MVC.

Jetstream: kit de inicio de aplicaciones bellamente diseñado para Laravel y proporciona el punto de partida perfecto para su próxima aplicación de 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.

Bootstrap: Framework de front-end basado en CSS desarrollado por Twitter en 2010, facilita los tiempos de desarrollo para el diseño de un sitio.

AdminLTE: Panel de administración para Bootstrap creado por el estudio Almsaeed. Es una solución de código abierto basada en un diseño modular que permite una construcción y personalización sencillas.

Laragon: Entorno de desarrollo universal portátil, aislado, rápido y potente para PHP, Node.js, Python, Java, Go, Ruby. Es rápido, liviano, fácil de usar y fácil de extender.

PhpMyAdmin: Aplicación web que sirve para administrar bases de datos MySQL de forma sencilla y con una interfaz amistosa. Se trata de un software muy popular basado en PHP. La ventaja de usar una aplicación web es que nos permite conectarnos con servidores remotos, a los cuales no siempre se puede acceder usando programas de interfaz gráfica.

MVC: (Modelo, Vista y controlador) es un Patrón en el diseño de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Enfatiza una separación entre la lógica de negocios y su visualización.

Servidor web: Un servidor web o servidor HTTP es un programa informático que procesa una aplicación del lado del servidor, realizando conexiones bidireccionales o unidireccionales y síncronas o asíncronas con el cliente y generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente.

Scrum: Marco de trabajo para desarrollo ágil de software que se ha expandido a otras industrias. Es un proceso en el que se aplican de manera regular un conjunto de buenas

prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de proyectos.

Ajax: (Asynchronous JavaScript and XML) se refiere a un grupo de tecnologías que se utilizan para desarrollar aplicaciones web. Al combinar estas tecnologías, las páginas web parecen ser más receptivas puesto que los paquetes pequeños de datos se intercambian con el servidor y las páginas web no se vuelven a cargar cada vez que un usuario realiza un cambio de entrada.