UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Sistema Web de Administración de Actividades y Personal de una Empresa Dedicada al Ramo de Transporte Público

INFORME DE ACTIVIDADES PROFESIONALES

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Rodrigo Abdiel Puente Estrada

ASESOR DE INFORME

M.C.I. Jorge Alberto Solano Gálvez

Ciudad Universitaria, Cd. Mx., 2026

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado SISTEMA WEB DE ADMINISTRACION DE ACTIVIDADES Y PERSONAL DE UNA EMPRESA DEDICADA AL RAMO DE TRANSPORTE PUBLICO que presenté para obtener el titulo de INGENIERO EN COMPUTACIÓN es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

RODRIGO ABDIEL PUENTE ESTRADA
Número de cuenta: 318126218

Tabla de contenido

Capítulo I. La Empresa

Introducción

En este capítulo se presenta una descripción general de la empresa "Corredor", una organización mexicana dedicada al transporte público que, a lo largo de varias décadas, ha desempeñado un papel fundamental en la movilidad dentro del sur de la Ciudad de México. Este apartado expone su origen, evolución e importancia dentro del sector, con el propósito de contextualizar adecuadamente su misión, visión y la problemática que dio lugar al desarrollo del sistema administrativo que se plantea en este trabajo. Comprender la naturaleza y trayectoria de la empresa es esencial para entender las necesidades específicas que motivaron la creación de una solución tecnológica capaz de centralizar y optimizar la gestión de sus procesos internos.

1.1 Contexto de la empresa

La empresa Corredor es una organización mexicana dedicada al transporte público desde la década de 1980. A lo largo de más de treinta años de operación, se ha consolidado como un pilar fundamental en la movilidad de los habitantes de las alcaldías Milpa Alta, Xochimilco, Tlalpan y Coyoacán. Su trayectoria comenzó con vehículos particulares; posteriormente incorporó combis, más tarde microbuses y, finalmente, obtuvo la concesión de un corredor de transporte, lo que marcó su transición de un modelo basado en concesionarios individuales a una estructura empresarial compuesta por accionistas. En su etapa más reciente, la empresa incorporó autobuses como unidades principales para la operación del corredor.

Actualmente, el Corredor cuenta con una plantilla superior a los 300 empleados, entre operadores, personal administrativo, mantenimiento y supervisión, y moviliza a más de cien mil pasajeros cada semana, lo que representa una base significativa de clientes recurrentes. Su cobertura se extiende a través de vialidades principales como la carretera a Oaxtepec, Prolongación División del Norte, Calzada de Miramontes y Avenida Taxqueña. Gracias a esta labor, la empresa se ha convertido en un actor clave en la conectividad entre la periferia y zonas urbanas más céntricas de la Ciudad de México, contribuyendo al desarrollo social y económico de las comunidades que atiende.

1.2 Misión y visión

Antes de presentar los elementos rectores del Corredor, es importante señalar que la misión y visión constituyen los principios orientadores que guían sus acciones operativas, administrativas y estratégicas. Ambos conceptos reflejan el compromiso con los usuarios y con la mejora constante del servicio público que ofrece.

Misión

La misión de Corredor es inspirar confianza mediante un servicio de transporte seguro, utilizando unidades en óptimas condiciones y operando en las principales avenidas del sur de la Ciudad de México, para ofrecer a los usuarios un traslado confiable, cómodo y seguro hacia su destino.[1]

Visión

La visión de Corredor es convertirse en una organización de referencia en el transporte público terrestre de pasajeros, distinguida por su calidad, puntualidad, seguridad y comodidad, esforzándose siempre por satisfacer plenamente las expectativas de sus clientes y garantizar una experiencia de viaje excepcional.[1]

1.3 Problemática

Para comprender la necesidad del sistema web propuesto, es importante analizar la forma en que la información era gestionada dentro del Corredor antes de la implementación de una solución tecnológica. Desde sus inicios, gran parte de los procesos administrativos se realizaban mediante registros manuales en papel, cuadernos y hojas de cálculo, lo que generaba múltiples problemas relacionados con la conservación, disponibilidad y confiabilidad de la información.

En el área de jornadas laborales, los operadores registraban sus horas de trabajo en formatos impresos que eran almacenados dentro de las unidades de transporte. Esta práctica ocasionaba con frecuencia la pérdida o extravío de documentos, ya que las hojas podían ser olvidadas, dañadas o desechadas accidentalmente. Durante temporadas de lluvia, era común que los formatos se mojaran y deterioraran debido a la humedad dentro de los autobuses, provocando que la información se volviera ilegible o se perdiera por completo. Cuando esto ocurría, era necesario volver a elaborar los registros o depender únicamente de la información proporcionada verbalmente por los operadores, sin posibilidad de verificar con exactitud las horas realmente laboradas.

De manera similar, en el área de mantenimiento las reparaciones y servicios realizados a las unidades eran registrados en cuadernos físicos. La pérdida, deterioro o extravío de estos cuadernos ocasionaba la desaparición de información histórica importante sobre las intervenciones realizadas a los vehículos. Como consecuencia, resultaba difícil determinar con precisión los costos acumulados de mantenimiento, identificar patrones de fallas o planificar adecuadamente los presupuestos destinados a las unidades.

En el departamento de siniestros no existía un sistema formal para almacenar el historial de accidentes o incidentes ocurridos. La evidencia se conservaba principalmente mediante fotografías almacenadas en teléfonos móviles del personal responsable. Esta situación representaba un riesgo considerable, ya que la información podía perderse por fallas en los dispositivos, eliminación accidental de archivos o falta de espacio de almacenamiento. Además, la ausencia de un repositorio centralizado dificultaba la consulta y seguimiento de eventos ocurridos anteriormente.

En el área de recaudo, los ingresos diarios generados por cada unidad eran anotados manualmente en cuadernos y posteriormente transcritos a hojas de cálculo. Este proceso incrementaba la probabilidad de errores de captura, inconsistencias en los datos y dificultades para interpretar la información financiera. En muchos casos, los archivos de cálculo no eran almacenados correctamente o se guardaban en ubicaciones diferentes, provocando que se perdieran o que únicamente se conservaran versiones incompletas de los registros.

Asimismo, en diversas áreas administrativas se utilizaban hojas de cálculo como herramienta complementaria para organizar información. Sin embargo, la falta de procedimientos estandarizados

para el almacenamiento y respaldo de los archivos ocasionaba pérdidas de información, duplicidad de registros y versiones inconsistentes de un mismo documento. Estas situaciones afectaban directamente la calidad de los análisis realizados por la empresa y limitaban la capacidad de tomar decisiones fundamentadas en información confiable.

La dependencia de registros físicos y documentos dispersos generaba una administración fragmentada, dificultando la consulta rápida de información histórica, el seguimiento de procesos y la generación de reportes. Como resultado, la dirección de la empresa enfrentaba dificultades para evaluar el desempeño operativo, estimar presupuestos de mantenimiento, verificar jornadas laborales, controlar ingresos y dar seguimiento a incidencias relevantes.

Ante esta situación, se identificó la necesidad de implementar una herramienta tecnológica que permitiera centralizar la información de todas las áreas en una única plataforma, garantizando la integridad, disponibilidad y seguridad de los datos. Mediante un sistema web de gestión administrativa, la empresa podría reducir la pérdida y duplicidad de información, agilizar la consulta de registros históricos y mejorar significativamente la toma de decisiones basada en datos confiables y actualizados.

1.4 Organigrama y contexto de la participación profesional

En la **Figura 1** se presenta el organigrama general del Corredor, el cual muestra la estructura jerárquica de la organización y la distribución de las áreas que dependen directamente de la Dirección General. Este organigrama permite identificar la relación existente entre los distintos departamentos y ubicar el área de Sistemas de Tecnologías de la Información dentro de la estructura administrativa de la empresa.

El Corredor cuenta con una organización conformada por diversas áreas encargadas de garantizar la operación diaria del servicio de transporte. En la parte superior de la estructura se encuentra la Dirección General, responsable de la supervisión de todas las actividades administrativas y operativas de la empresa, así como de la toma de decisiones estratégicas.

Entre las áreas que dependen directamente de la Dirección General se encuentran Recaudo, Sistemas de Tecnologías de la Información (TI), Diésel, Operaciones, Recursos Humanos, Monitoreo y Telemetría, y Siniestros, cada una con funciones específicas para el correcto funcionamiento de la organización.

El departamento de Recaudo era responsable de la recuperación y control de los ingresos generados por las unidades de transporte. Durante el turno nocturno, el personal recorría todas las unidades para retirar físicamente el dinero acumulado en las alcancías de los autobuses. Posteriormente, las monedas eran procesadas mediante una máquina contadora que generaba un comprobante impreso con la cantidad total contabilizada. Esta información era registrada manualmente en cuadernos de ingresos diarios y, en algunos casos, trasladada posteriormente a hojas de cálculo. Este proceso dependía completamente de registros físicos, lo que dificultaba la consulta histórica y aumentaba el riesgo de errores o pérdidas de información.

El área de Diésel tenía como función principal el control del combustible suministrado a las unidades. Cada tercer día, durante el turno nocturno, se realizaba el abastecimiento de combustible y se registraban manualmente datos como el número económico de la unidad, los litros suministrados y el nivel de combustible antes y después de la carga. Estos registros se realizaban en formatos impresos con tablas diseñadas para tal fin. Debido a que el proceso se desarrollaba en áreas abiertas y expuestas a condiciones ambientales adversas, las hojas podían deteriorarse por humedad, suciedad o contacto con sustancias derivadas de la operación, comprometiendo la integridad de la información registrada.

El departamento de Operaciones era el encargado de coordinar la actividad diaria de los operadores y las unidades en servicio. Sin embargo, no contaba con herramientas tecnológicas que permitieran monitorear en tiempo real el cumplimiento de las jornadas laborales ni conocer con precisión el estado operativo de cada conductor. La comunicación se realizaba principalmente mediante radiofrecuencia, y cuando ocurría algún incidente o siniestro, el seguimiento dependía de la información transmitida por radio y de las grabaciones obtenidas mediante sistemas de videovigilancia proporcionados por terceros. Esta situación dificultaba la supervisión eficiente de las operaciones y limitaba la capacidad de generar registros confiables sobre los eventos ocurridos durante el servicio.

El área de Recursos Humanos gestionaba toda la información relacionada con los operadores y el personal administrativo. Los expedientes de los empleados, incluyendo documentación personal, fotografías, registros de ingreso, reportes disciplinarios y demás información laboral, se almacenaban físicamente en carpetas. La búsqueda de información requería la revisión manual de archivos, lo que hacía lento el acceso a los datos. Asimismo, el cálculo de la nómina dependía de los registros de jornadas laborales elaborados en papel, los cuales frecuentemente debían buscarse en bodegas, oficinas o incluso dentro de las unidades de transporte. Adicionalmente, al tratarse de documentos escritos a mano, en numerosas ocasiones la legibilidad de la información era limitada, generando errores y retrasos en los procesos administrativos.

El departamento de Monitoreo y Telemetría utilizaba principalmente un sistema de videovigilancia proporcionado por un proveedor externo, el cual permitía visualizar en tiempo real las cámaras instaladas en las unidades. A partir de estas observaciones se elaboraban reportes relacionados con faltas operativas o incumplimientos por parte de los operadores. Sin embargo, dichos reportes eran almacenados en documentos de texto, hojas impresas o archivos de cálculo sin una estructura centralizada, lo que ocasionaba duplicidad de información, pérdida de registros y dificultades para dar seguimiento a incidentes ocurridos con anterioridad.

Por su parte, el departamento de Siniestros no contaba con un sistema formal para registrar accidentes o incidentes. La documentación se limitaba principalmente a fotografías tomadas con teléfonos celulares del personal responsable. No existían expedientes digitales ni mecanismos de seguimiento que permitieran consultar el historial de eventos ocurridos, identificar patrones o dar continuidad a los procesos derivados de cada incidente. Esta situación representaba un riesgo importante debido a la posible pérdida de evidencia por daños en los dispositivos, eliminación accidental de archivos o falta de almacenamiento.

Mi participación profesional se desarrolló dentro del departamento de Sistemas de Tecnologías de la Información, donde desempeñé el cargo de Jefe de Sistemas TI. Inicialmente, mis responsabilidades estaban orientadas al soporte técnico, administración de cuentas de correo institucionales, gestión de

licencias de software mediante los servicios empresariales contratados con Microsoft y mantenimiento de la infraestructura tecnológica de la organización. Asimismo, era responsable de garantizar la conectividad de internet, la operación de la red informática y el mantenimiento preventivo y correctivo de los equipos de cómputo.

Como parte de las actividades realizadas dentro del área de TI, también participé en la adquisición y configuración de un dominio institucional, así como en la implementación de una página web informativa hospedada mediante servicios comerciales de Hostinger. Adicionalmente, realicé las configuraciones necesarias para integrar el servicio de correo electrónico corporativo con el dominio adquirido por la empresa.

El desempeño de estas funciones me permitió conocer de manera directa la operación de cada departamento e identificar diversas problemáticas relacionadas con el manejo manual de la información, la falta de centralización de los registros, la duplicidad de datos y la dificultad para acceder oportunamente a información confiable. A partir de esta experiencia surgió la iniciativa de analizar, diseñar y desarrollar un sistema web administrativo que permitiera integrar la información de todas las áreas en una única plataforma, optimizando los procesos internos y mejorando la capacidad de la empresa para tomar decisiones basadas en datos precisos y actualizados.

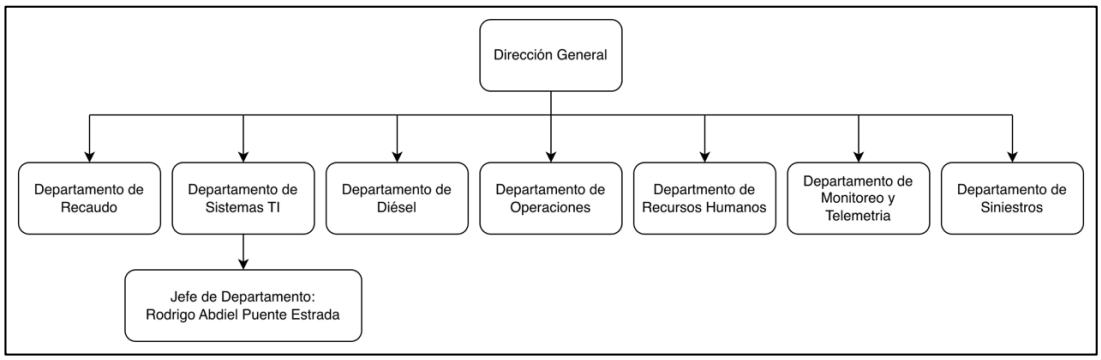

Figura 1.1 Organigrama general del Corredor.

Capítulo II. Marco teórico

Introducción

El desarrollo de un sistema web administrativo para una empresa de transporte público requiere apoyarse en conceptos y herramientas propias de los sistemas de información y del desarrollo de software. En este capítulo se presenta el marco teórico que sustenta el proyecto, abordando la importancia de los sistemas de información en el transporte urbano, el papel de las aplicaciones web como solución para la gestión administrativa y, finalmente, las tecnologías utilizadas para implementar la plataforma. Este capítulo permite comprender por qué una solución web resulta pertinente para el contexto del Corredor y cómo las tecnologías elegidas contribuyen a la centralización, disponibilidad y confiabilidad de la información.

2.1 Sistemas de información en el transporte público

Los sistemas de información desempeñan un papel fundamental en la administración moderna de las organizaciones, ya que permiten centralizar datos, automatizar procesos y proporcionar información confiable para la toma de decisiones. Sin embargo, en el sector del transporte público las necesidades administrativas y operativas suelen ser más complejas debido a la interacción de múltiples áreas, como la gestión de operadores, control de unidades, mantenimiento, abastecimiento de combustible, recaudo, monitoreo y seguimiento de incidencias.

Durante el análisis previo al desarrollo del presente proyecto, se identificó que las soluciones comerciales ofrecidas a la empresa estaban orientadas principalmente a la administración de recursos humanos. Estos sistemas permitían controlar aspectos como asistencia, faltas, retardos, jornadas laborales y algunos procesos relacionados con la nómina. No obstante, dichas herramientas no contemplaban la operación integral de una empresa de transporte público, por lo que resultaban insuficientes para cubrir las necesidades reales del Corredor.

Entre las principales limitaciones detectadas se encontraba la ausencia de módulos para el control de recaudo, administración de combustible, seguimiento de mantenimiento de unidades, registro de siniestros, monitoreo operativo y gestión documental de los diferentes departamentos. Esto implicaba que, aun adquiriendo alguno de estos sistemas, gran parte de los procesos continuarían realizándose de forma manual o mediante herramientas independientes, manteniendo los problemas de dispersión y duplicidad de información.

Asimismo, durante la investigación realizada se observó que actualmente no existen soluciones especializadas para la administración integral de empresas de transporte público urbano en México. Particularmente en la Ciudad de México y su zona metropolitana, la adopción de plataformas tecnológicas orientadas a la gestión administrativa de corredores de transporte aún se encuentra en etapas iniciales, y no se identificaron proveedores que ofrecieran una solución capaz de integrar todas las áreas operativas y administrativas requeridas por la organización.

Ante esta situación, se concluyó que la alternativa más adecuada era el desarrollo de un sistema propio, diseñado específicamente para las necesidades y procesos internos del Corredor. Esta decisión permitiría construir una plataforma adaptada a la operación real de la empresa, integrando en un mismo entorno la información de Recursos Humanos, Operaciones, Recaudo, Diésel, Monitoreo, Siniestros y demás áreas involucradas, logrando así una administración más eficiente, centralizada y alineada con los requerimientos específicos de la organización.

2.2 Aplicaciones web en la gestión administrativa

Las aplicaciones web se han consolidado como una de las soluciones más prácticas y escalables para la gestión administrativa en organizaciones de distintos sectores. A diferencia de los sistemas *stand-alone* instalados en un solo equipo, una aplicación web puede consultarse desde cualquier dispositivo con acceso a internet, lo cual facilita el trabajo distribuido y el acceso remoto a la información. Al operar sobre un sistema centralizado, la actualización de datos ocurre en un único lugar, evitando duplicidades y reduciendo inconsistencias entre archivos y registros.

Otra ventaja relevante es la disminución de costos de mantenimiento, ya que no es necesario realizar instalaciones complejas en cada dispositivo, y las actualizaciones pueden aplicarse directamente al servidor. Así mismo, su disponibilidad tiende a ser mayor, ya que el servicio permanece accesible mientras el servidor se encuentre activo, y su escalabilidad permite integrar nuevos módulos o funcionalidades conforme evolucionan las necesidades de la organización.

En el caso particular del transporte público, estas ventajas se traducen en un mejor control de los recursos humanos y materiales, mayor rapidez en procesos administrativos como el registro de jornadas o consulta de incidencias y la posibilidad de incorporar mejoras futuras sin reconstruir toda la solución. En consecuencia, una plataforma web representa una alternativa pertinente para modernizar la administración de corredores urbanos y reducir errores derivados del manejo manual de la información

2.3 Fundamentos tecnológicos del desarrollo web

El desarrollo de aplicaciones web administrativas se sustenta en diversos conceptos tecnológicos que permiten estructurar la información, procesarla en un servidor y presentarla al usuario a través de una interfaz accesible desde internet. Estos fundamentos permiten comprender el funcionamiento interno del sistema más allá de las herramientas específicas utilizadas para su implementación (*Sommerville*, 2016).

2.3.1 Servidor web

Un servidor web es el componente encargado de recibir las solicitudes realizadas por los usuarios a través de un navegador y devolver como respuesta los recursos solicitados mediante el protocolo HTTP. Este modelo se basa en una arquitectura cliente-servidor, donde las peticiones enviadas por el usuario son procesadas por el servidor antes de generar una respuesta adecuada. (Mozilla Foundation, s.f.).

En aplicaciones administrativas, el servidor web no solo distribuye contenido estático, sino que también ejecuta procesos que permiten acceder a bases de datos, procesar información y generar contenido dinámico de acuerdo con las necesidades de los usuarios.

Para el desarrollo e implementación del sistema se seleccionó Internet Information Services (IIS), el servidor web desarrollado por Microsoft e incluido de forma nativa en los sistemas operativos Windows Server y en diversas versiones de Windows. La elección de esta tecnología se fundamentó principalmente en que la infraestructura tecnológica de la empresa se encontraba basada en sistemas operativos Windows, lo que permitía aprovechar una solución integrada al propio entorno de trabajo.

Al formar parte del ecosistema de Microsoft, IIS ofrece una integración directa con los servicios y herramientas del sistema operativo, facilitando las tareas de instalación, configuración, administración y mantenimiento del servidor. Asimismo, esta integración permite optimizar el uso de los recursos del sistema, proporcionando un mejor rendimiento y estabilidad en comparación con soluciones que requieren capas adicionales de compatibilidad o configuraciones externas.

Otra ventaja relevante de IIS es la disponibilidad de herramientas administrativas incorporadas que permiten gestionar sitios web, certificados de seguridad, permisos de acceso y aplicaciones desde una interfaz centralizada. Esto simplifica las labores de administración y reduce la complejidad operativa para el personal encargado de la infraestructura tecnológica.

Por estas razones, IIS fue seleccionado como plataforma de alojamiento para el sistema administrativo desarrollado, proporcionando un entorno estable, eficiente y compatible con la infraestructura tecnológica existente dentro de la organización.

2.3.2 Lenguajes de programación del lado del servidor

Los lenguajes de programación del lado del servidor permiten implementar la lógica de una aplicación web, procesar información enviada por los usuarios y gestionar la comunicación con los sistemas de almacenamiento de datos. Estos lenguajes se ejecutan directamente en el servidor, donde procesan las solicitudes recibidas y generan las respuestas que posteriormente son enviadas al navegador del usuario.

Entre los lenguajes más utilizados para el desarrollo de aplicaciones web se encuentra PHP, el cual permite interactuar con diferentes sistemas gestores de bases de datos. Su amplia compatibilidad con servidores web y su facilidad de implementación lo convierten en una alternativa ampliamente utilizada para el desarrollo de aplicaciones empresariales.

Una de las principales funciones de PHP consiste en generar y ejecutar sentencias SQL (Structured Query Language), las cuales son utilizadas para establecer comunicación con las bases de datos. Mediante estas sentencias es posible realizar operaciones de consulta, inserción, actualización y eliminación de información almacenada en los sistemas gestores de bases de datos.

La combinación entre PHP y SQL permite que las aplicaciones web administren información de manera eficiente, garantizando la interacción entre la interfaz de usuario y los datos almacenados.

Además, el lenguaje proporciona mecanismos para validar información, controlar sesiones de usuario y aplicar reglas de negocio antes de ejecutar operaciones sobre la base de datos.

Desde el enfoque de la ingeniería de software, la lógica implementada en el servidor es fundamental para asegurar el correcto procesamiento de la información, la integridad de los datos y el funcionamiento adecuado de las aplicaciones web modernas (Pressman & Maxim, 2020).

2.3.3 Lenguajes de marcado y presentación

Los lenguajes de marcado y presentación son fundamentales en el desarrollo de aplicaciones web, ya que permiten definir tanto la estructura de los contenidos como su apariencia visual dentro del navegador. Estos estándares facilitan la construcción de interfaces organizadas, accesibles y compatibles con diferentes dispositivos y navegadores web (World Wide Web Consortium [W3C], s.f.).

En el desarrollo del sistema se utilizó **HTML5** como lenguaje de marcado para construir la estructura de las páginas web. Mediante HTML5 se desarrollaron formularios de captura, tablas de consulta, menús de navegación, ventanas modales y diversos componentes de la interfaz utilizados por los usuarios. La elección de HTML5 se realizó debido a su amplia compatibilidad con navegadores modernos, su integración nativa con JavaScript y CSS3, y las capacidades de validación incorporadas en los elementos de formularios.

Por su parte, **CSS3** fue utilizado para definir la presentación visual del sistema. A través de hojas de estilo se implementaron aspectos relacionados con la distribución de los elementos en pantalla, tipografías, colores, botones, tablas, formularios y componentes gráficos utilizados en la plataforma. Asimismo, CSS3 permitió desarrollar interfaces adaptables a diferentes resoluciones de pantalla mediante técnicas de diseño responsivo, mejorando la experiencia de uso para el personal administrativo.

La combinación de HTML5 y CSS3 permitió construir una interfaz visual organizada y consistente para todos los módulos del sistema, facilitando la interacción de los usuarios con la plataforma y contribuyendo a una mejor experiencia durante la captura, consulta y administración de la información.

2.3.4 Lenguajes de programación del lado del cliente

Los lenguajes ejecutados en el navegador permiten agregar interactividad a la aplicación web. Entre sus funciones se encuentran la validación preliminar de formularios, la manipulación dinámica del contenido y la mejora de la experiencia del usuario sin necesidad de recargar completamente la página (Mozilla Foundation, s.f.).

En el desarrollo del sistema se utilizó **JavaScript** como principal lenguaje de programación del lado del cliente. Su implementación permitió gestionar la interacción entre los usuarios y la plataforma, realizando validaciones preliminares de los datos capturados antes de enviarlos al servidor y reduciendo así el número de solicitudes innecesarias hacia la base de datos.

Asimismo, JavaScript fue utilizado para la manipulación dinámica de los elementos de la interfaz, permitiendo mostrar u ocultar formularios, ventanas modales, tablas, menús y diferentes componentes de la plataforma de acuerdo con las acciones realizadas por el usuario. Esta capacidad contribuyó a mejorar la usabilidad del sistema y a ofrecer una experiencia más fluida durante la navegación.

Otra de las funciones principales de JavaScript dentro de la arquitectura desarrollada fue la comunicación asíncrona con el servidor mediante solicitudes HTTP. A través de estas peticiones, la información capturada en los formularios es enviada a archivos PHP encargados de procesar la lógica de negocio y ejecutar las operaciones correspondientes sobre la base de datos PostgreSQL. Posteriormente, las respuestas son recibidas en formato JSON y procesadas nuevamente por JavaScript para actualizar la información mostrada en pantalla sin necesidad de recargar la página completa.

Además, JavaScript permitió implementar funcionalidades avanzadas utilizadas en diversos módulos del sistema, como temporizadores de jornadas laborales, validación de formularios, filtros dinámicos de búsqueda, generación de tablas interactivas, geolocalización mediante OpenStreetMap, actualización periódica de información operativa y administración de eventos dentro de la interfaz.

La utilización de JavaScript permitió desarrollar una aplicación más dinámica, eficiente y responsive, facilitando la interacción entre el usuario, el servidor web y la base de datos dentro de la arquitectura cliente-servidor implementada.

2.3.5 Entornos de despliegue

El entorno de despliegue corresponde al conjunto de recursos de hardware y software necesarios para alojar y ejecutar una aplicación informática de manera estable y accesible para los usuarios. Una adecuada configuración de estos entornos permite garantizar la disponibilidad de los servicios, facilitar el mantenimiento de la plataforma y asegurar el correcto funcionamiento de los componentes que integran el sistema (Pressman & Maxim, 2020).

Durante el desarrollo de este proyecto no existían dentro de la empresa áreas especializadas encargadas de las fases de desarrollo, validación, pruebas y liberación de software. Debido a la estructura del departamento de Tecnologías de la Información y a mis funciones como responsable del área, estas actividades fueron realizadas directamente por mí.

El proceso de trabajo consistió en desarrollar nuevas funcionalidades o modificaciones al sistema, realizar las pruebas necesarias para verificar su correcto funcionamiento y, posteriormente, implementar los cambios en el entorno de producción. Este esquema me permitió mantener un control directo sobre cada etapa del desarrollo, asegurando que las funcionalidades cumplieran con los requerimientos establecidos antes de ser utilizadas por los usuarios finales.

Para la implementación de la plataforma opté por utilizar una arquitectura de despliegue basada en servidores independientes para cada servicio principal. El servidor web encargado de alojar la aplicación se encuentra separado del servidor que almacena la base de datos. Esta decisión fue tomada con el objetivo de evitar que ambos servicios compitieran por los mismos recursos de hardware, como memoria, almacenamiento y capacidad de procesamiento.

La separación de servicios permitió distribuir de manera más eficiente la carga de trabajo entre los servidores, mejorando el rendimiento general de la infraestructura. Además, esta arquitectura facilita las tareas de administración, mantenimiento y crecimiento futuro del sistema, ya que cada componente puede ser gestionado de manera independiente sin afectar el funcionamiento de los demás servicios.

Asimismo, la utilización de servidores dedicados para cada función contribuye a reducir posibles problemas de rendimiento derivados del incremento en el número de usuarios o del crecimiento de la información almacenada. De esta manera, la infraestructura implementada proporciona una base más estable, escalable y preparada para futuras ampliaciones de la plataforma administrativa desarrollada.

2.4 Metodología utilizada e ingeniería de software

La metodología de desarrollo de software define el conjunto de enfoques, procesos y técnicas que se utilizan para planificar, desarrollar, implementar y mantener un sistema de información de manera estructurada y controlada. De acuerdo con *Pressman y Maxim* (2020), una metodología proporciona un marco de trabajo que permite organizar las actividades del desarrollo, reducir riesgos y asegurar la calidad del producto final.

En este proyecto se adoptó una **metodología de tipo incremental**, la cual pertenece a los modelos evolutivos del desarrollo de *software*. Este enfoque se basa en la construcción del sistema mediante incrementos o módulos funcionales, donde cada incremento agrega nuevas funcionalidades al sistema previamente desarrollado y validado. Según *Sommerville* (2016), el desarrollo incremental permite entregar versiones operativas del software de forma temprana, facilitando la retroalimentación del usuario y la adaptación a cambios en los requerimientos.

La metodología incremental se apoya en principios fundamentales de la **ingeniería de *software***, disciplina que integra métodos, herramientas y buenas prácticas para desarrollar sistemas confiables, mantenibles y alineados con necesidades reales (IEEE, 2014). Bajo este enfoque, el desarrollo se realiza de manera progresiva, priorizando la calidad, la validación continua y la correcta documentación del sistema.

En concordancia con este modelo, el proceso de desarrollo se estructura generalmente en las siguientes etapas:

a) **Levantamiento y análisis de requerimientos:** identificación de los procesos administrativos, necesidades de control y definición de los requerimientos funcionales del sistema.
b) **Análisis y diseño del sistema:** definición de los módulos, reglas de negocio y estructura de la información.
c) **Implementación por módulos:** desarrollo *Fullstack*, abarcando tanto la programación del lado del servidor como la interfaz web.
d) **Pruebas y validación:** ejecución de pruebas funcionales para verificar el correcto registro, consulta y consistencia de los datos.
e) **Documentación básica:** elaboración de documentación que describe el funcionamiento general del sistema.

La aplicación de esta metodología permitió un mayor control del avance del proyecto, la detección temprana de errores y la entrega progresiva de funcionalidades operativas, asegurando que el sistema desarrollado cumpliera con los objetivos establecidos y con los principios de calidad propios de la ingeniería de software.

2.5 Estructuras de datos

Las estructuras de datos permiten organizar y manipular información de forma eficiente dentro de un sistema. En aplicaciones administrativas, su uso es esencial para manejar colecciones de registros, resultados de consultas y datos capturados en formularios.

En el sistema desarrollado se manejan estructuras como **arreglos** y **listas**, utilizadas para almacenar temporalmente resultados provenientes de la base de datos y construir tablas dinámicas de consulta. Asimismo, se trabajó con **registros** (conjuntos de atributos) que representan entidades (Tipos de datos abstractos) como empleados, unidades y jornadas, permitiendo un manejo ordenado de la información.

2.6 Algoritmos

Un algoritmo es un conjunto de pasos ordenados que permite resolver un problema o ejecutar una tarea. En este sistema, los algoritmos se aplicaron principalmente para:

- **Validación de formularios**, asegurando que los datos cumplieran reglas mínimas antes de almacenarse.
- **Procesamiento de registros**, como inserción, actualización y consulta en la base de datos.
- **Generación de resultados**, organizando la información en tablas o reportes para el usuario.

El uso de algoritmos claros y consistentes permite reducir errores, mejorar la confiabilidad del sistema y mantener una lógica de operación comprensible.

2.7 Bases de datos

Una base de datos es un conjunto organizado de información que puede ser consultada, actualizada y administrada de forma eficiente. En los sistemas administrativos, las bases de datos permiten centralizar la información, mantener la integridad de los datos y facilitar la generación de reportes confiables que apoyan la toma de decisiones.

El **modelo relacional** organiza la información en tablas relacionadas entre sí mediante **llaves primarias y foráneas**, lo que permite reducir la duplicidad de datos, mejorar la consistencia de la información y asegurar reglas de integridad referencial. Para este proyecto se adoptó este enfoque debido a la necesidad de almacenar información estructurada, como personal, unidades, jornadas laborales e incidencias, manteniendo relaciones claras entre los registros.

Dentro del diseño de bases de datos relacionales, la **normalización** es un proceso fundamental que consiste en organizar las tablas y sus atributos con el objetivo de minimizar redundancias, evitar anomalías en operaciones de inserción, actualización y eliminación, y asegurar la coherencia de la información almacenada.

El proceso de normalización se estructura en diferentes **formas normales**, entre las cuales destacan:

- **Primera Forma Normal (1FN)**: establece que cada tabla debe contener atributos con valores atómicos, es decir, sin grupos repetitivos ni atributos multivaluados, y que cada registro pueda identificarse de manera única mediante una clave primaria.
- **Segunda Forma Normal (2FN)**: se alcanza cuando la tabla se encuentra en 1FN y todos los atributos no clave dependen completamente de la clave primaria, eliminando dependencias parciales en tablas con claves compuestas.
- **Tercera Forma Normal (3FN)**: se cumple cuando la tabla está en 2FN y no existen dependencias transitivas, es decir, cuando los atributos no clave dependen únicamente de la clave primaria y no de otros atributos no clave.

La aplicación de estas formas normales permite diseñar bases de datos más consistentes, fáciles de mantener y escalables. En sistemas administrativos, como el desarrollado en este proyecto, la normalización resulta especialmente importante debido al manejo de grandes volúmenes de información y a la necesidad de garantizar la integridad de los datos a lo largo del tiempo.

Un sistema gestor de bases de datos (SGBD) es un software especializado en almacenar, organizar y administrar grandes volúmenes de información estructurada. Permite realizar operaciones como inserción, actualización y consulta de datos, asegurando integridad, seguridad y consistencia (IBM, s.f.).

Desde una perspectiva de ingeniería de software, el uso de un SGBD permite aplicar principios de organización estructurada de datos y mantener control sobre la información persistente del sistema (Sommerville, 2016).

2.8 Programación orientada a objetos

La Programación Orientada a Objetos (POO) es un paradigma que organiza el *software* a partir de entidades (objetos) que representan elementos del mundo real. Sus principios son **encapsulamiento, abstracción, herencia y polimorfismo**.

Aunque el sistema es web, se aplicaron principios de modularidad y organización por entidades, facilitando el mantenimiento y la expansión futura del sistema. Este enfoque permite que el desarrollo sea más estructurado, evitando código duplicado y mejorando la claridad del proyecto.

2.9 Administración de proyectos de software

La administración de proyectos de software se enfoca en planificar, organizar y controlar actividades para cumplir objetivos dentro de un tiempo y recursos definidos. En este proyecto se consideró un control por etapas, donde se definieron entregables por módulo y se dio seguimiento a su implementación.

Esto permitió mantener orden en el desarrollo, priorizar funciones críticas y asegurar que el sistema respondiera a las necesidades administrativas observadas.

2.10 Arquitectura cliente-servidor

La arquitectura cliente-servidor consiste en la separación entre:

- **Cliente:** interfaz que utiliza el usuario (navegador web).
- **Servidor:** donde se procesa la lógica del sistema y se conecta a la base de datos.

Este enfoque permite centralizar la información y las reglas de negocio, asegurando que los cambios y actualizaciones se realicen en un solo punto.

2.11 Seguridad y control de acceso

En sistemas administrativos es indispensable proteger la información mediante mecanismos de autenticación y control de permisos. La seguridad se fortalece mediante:

- credenciales de acceso,
- validación de entradas,
- control de sesiones,
- restricción de acciones según el rol del usuario,
- entre otros.

Estos principios ayudan a prevenir accesos no autorizados y mantener la integridad de los datos.

2.12 Tecnologías utilizadas en el proyecto

La implementación del sistema web desarrollado para el Corredor se apoyó en un conjunto de tecnologías que permiten construir aplicaciones en un entorno cliente-servidor, garantizando el almacenamiento seguro de información y una interacción eficiente para el usuario final. Las herramientas seleccionadas responden tanto a criterios técnicos como a la viabilidad práctica de implementación, considerando además que se trata de tecnologías ampliamente utilizadas, con documentación extensa y soporte comunitario.

2.12.1 Internet Information Services (IIS) como servidor de despliegue

Internet Information Services (IIS) es un servidor web desarrollado por Microsoft para sistemas operativos Windows, diseñado para alojar, administrar y publicar aplicaciones web de forma segura y eficiente. IIS ofrece soporte para múltiples tecnologías de desarrollo web y permite la integración con PHP mediante el módulo FastCGI, proporcionando un entorno estable para la ejecución de aplicaciones dinámicas.

En el presente proyecto, IIS fue utilizado como servidor de despliegue de la aplicación web, permitiendo la publicación y acceso al sistema a través de la red institucional e Internet para usuarios autorizados. Su implementación facilitó la administración centralizada de los servicios web, la gestión

de la seguridad del sistema y la integración con la base de datos PostgreSQL utilizada por la aplicación. Asimismo, IIS proporcionó un entorno confiable para el funcionamiento de los diferentes módulos del sistema, garantizando la disponibilidad y el acceso a la información de manera eficiente.

2.12.2 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, ampliamente utilizado en entornos profesionales debido a su robustez, seguridad y soporte para transacciones. En el sistema desarrollado, PostgreSQL se empleó para almacenar y organizar la información administrativa relacionada con operadores, unidades, jornadas laborales, suministro de combustible, siniestros e incidencias. Su uso permitió garantizar consistencias e integridad en los datos, evitando errores derivados de duplicidades o capturas incompletas, y mejorando la disponibilidad de la información para consultas administrativas.

2.12.3 HTML (HyperText Markup Language)

HTML se utilizó para estructurar las páginas que conforman la plataforma definiendo los elementos principales de la interfaz, como formularios de registro, menús de navegación, tablas de consulta y contenedores para presentar la información. Este lenguaje permite organizar el contenido de manera clara y compatible con navegadores web estándar.

2.12.4 PHP (Hypertext Processor)

PHP fue el lenguaje principal de lado del servidor. Se utilizó para procesar las solicitudes realizadas por los usuarios, gestionar el flujo de la lógica del negocio y ejecutar operaciones sobre la base de datos en PostgreSQL. Gracias a su integración con Apache, PHP facilitó la conexión entre la interfaz de usuario y el almacenamiento de información, permitiendo además generar contenido dinámico según las necesidades de cada módulo.

2.12.5 JavaScript

JavaScript permite dotar de interactividad a la aplicación web, principalmente mediante la validación de los formularios antes de enviarlos al servidor, generación de mensajes dinámicos y mejora de la experiencia del usuario en las pantallas de captura y consulta, su uso también contribuye a realizar actualizaciones ágiles de contenido sin requerir recarga por completo la página, lo cual mejora el rendimiento percibido por el usuario y reduce fricción en tareas repetitivas.

2.12.6 CSS (Cascading Style Sheets)

CSS se empleó para definir el diseño visual y su maquetación responsiva. El enfoque adoptado buscó facilitar el mantenimiento y mejorar las consistencias de la interfaz, por lo que los estilos se organizaron en archivos separados por página, lo cual permite modificar secciones específicas sin afectar al resto del sistema, asimismo, se consideró un *layout* responsivo mediante el uso de estructuras como *flexbox* y *grid* para adaptar a paneles y tablas tanto en dispositivos móviles como en equipos de escritorio.

De manera complementaria, se mantuvo un sistema visual coherente mediante una paleta de colores consistente, tipografías legibles y espacios uniformes, buscando que los elementos se percibieran

homogéneos en todo el sitio. También se trabajó con componentes reutilizables -como botones, tablas y formularios- mediante clases que favorecen la uniformidad. En términos de accesibilidad básica, se produjo un contraste suficiente, estados de enfoque visibles y tamaños de fuente relativos. Finalmente, se consideraron aspectos de rendimiento en producción, como la minificación y una carga ordenada de estilos para evitar efectos visuales no deseados como *FOUC (Flash of Unstyled Content)*.

Capítulo III. El proyecto

Introducción

En el presente capítulo se expone el desarrollo del proyecto realizado para atender la problemática administrativa identificada en la empresa Corredor, dedicada al transporte público en la Ciudad de México. A partir del análisis de las necesidades operativas y administrativas de la organización, se diseñó e implementó un sistema web orientado a centralizar la información, optimizar los procesos internos y mejorar el acceso a los datos por parte de las distintas áreas involucradas.

El capítulo describe la forma en que la solución fue concebida desde la identificación de requerimientos hasta su implementación técnica, abordando la arquitectura general del sistema, el control de acceso por roles, la organización de los módulos, así como los principales elementos del *front-end*, *back-end* y base de datos. Asimismo, se presentan ejemplos representativos de la implementación, con el propósito de evidenciar el trabajo realizado y mostrar la correspondencia entre la problemática detectada y la solución desarrollada.

El objetivo de este capítulo es mostrar de manera estructurada cómo se llevó a cabo el análisis, diseño, desarrollo e integración del sistema web administrativo, destacando las decisiones técnicas adoptadas, el alcance funcional de la plataforma y su adecuación a los procesos reales de la empresa. De esta forma, se busca evidenciar que la solución propuesta no solo responde a una necesidad específica del entorno laboral, sino que constituye una herramienta tecnológica viable para fortalecer la gestión administrativa del Corredor

3.1 Descripción del proyecto y solución propuesta

El transporte público en la Ciudad de México constituye uno de los elementos fundamentales para la movilidad urbana, debido a la gran cantidad de usuarios que dependen diariamente de este servicio para trasladarse entre diferentes zonas de la ciudad. La operación de este sistema implica la coordinación de múltiples elementos, entre los que destacan la gestión del personal operativo, el control de unidades de transporte, el registro de jornadas laborales y el seguimiento de diversos procesos administrativos que garantizan la continuidad y calidad del servicio.

En el caso particular del Corredor, se identificó una problemática relacionada con la administración de la información operativa y administrativa. Diversos procesos, como el registro de operadores, el control de unidades, el seguimiento de jornadas laborales, el registro de incidencias y el control del suministro de diésel, se realizaban mediante formatos físicos o archivos de hojas de cálculo independientes. Esta forma de gestión dificultaba la centralización de la información, generaba duplicidad de registros y aumentaba el riesgo de inconsistencias o pérdida de datos.

Además, la información se encontraba dispersa entre diferentes áreas de trabajo, lo que complicaba su consulta o actualización de manera oportuna. Esta situación afectaba la eficiencia de los procesos administrativos y limitaba la disponibilidad de información confiable para la toma de decisiones dentro de la organización.

Ante este contexto, se propuso el desarrollo de un sistema web de gestión administrativa que permitiera centralizar la información en una plataforma digital accesible desde los equipos de la empresa. El objetivo principal de este sistema fue proporcionar una herramienta que facilitara el registro, consulta y actualización de la información administrativa relacionada con la operación del Corredor, contribuyendo a mejorar la organización de los procesos internos y la disponibilidad de los datos.

La solución planteada consistió en el análisis, diseño, desarrollo e implementación de una plataforma web que integrara distintos módulos administrativos dentro de un mismo sistema. De esta forma, los usuarios autorizados podrían acceder a la información desde un navegador web, registrar nuevos datos y consultar los registros existentes sin depender de documentos físicos o archivos dispersos.

El sistema fue desarrollado bajo una arquitectura cliente-servidor. En este modelo, el cliente corresponde al navegador web utilizado por los usuarios para interactuar con la plataforma, mientras que el servidor se encarga de procesar las solicitudes, ejecutar la lógica de negocio y realizar las operaciones necesarias sobre la base de datos.

Para la implementación del sistema se utilizaron tecnologías orientadas al desarrollo web y a la administración centralizada de información. El entorno de despliegue se configuró mediante Internet Information Services (IIS), servidor web de Microsoft utilizado para publicar y administrar la aplicación en un entorno Windows. La ejecución de la lógica del sistema se realizó mediante PHP, integrado a IIS a través de FastCGI, mientras que la base de datos se implementó utilizando PostgreSQL. Por su parte, la interfaz del sistema fue desarrollada con HTML, CSS y JavaScript, tecnologías ampliamente utilizadas para la construcción de aplicaciones web interactivas.

La elección de estas tecnologías se basó en su disponibilidad, estabilidad, facilidad de implementación y compatibilidad entre sí. Además, su uso permitió desarrollar una solución escalable, accesible desde navegador web y adecuada para las necesidades administrativas del Corredor.

Mediante la implementación de este sistema, fue posible estructurar y digitalizar diversos procesos administrativos del Corredor, permitiendo centralizar la información, mejorar la organización de los registros y facilitar el acceso controlado a los datos por parte del personal autorizado.

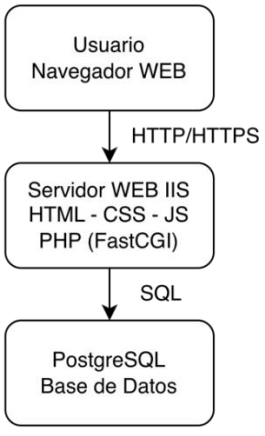

Figura 3.1 Arquitectura General del sistema administrativo

3.1.1 Levantamiento de requerimientos y análisis

El levantamiento de requerimientos constituyó una etapa fundamental para identificar las necesidades reales de la organización y definir las funcionalidades que debía integrar el sistema web administrativo. Esta actividad permitió comprender los procesos internos, reconocer a los usuarios involucrados y establecer las funciones necesarias para digitalizar y centralizar la información.

Durante esta etapa se realizaron reuniones con personal de las diferentes áreas administrativas y operativas del Corredor, con el objetivo de conocer sus actividades diarias, los registros que utilizaban y las dificultades que enfrentaban al consultar, actualizar o resguardar la información. Asimismo, se revisaron formatos impresos, hojas de cálculo y registros físicos empleados en los procesos cotidianos.

A partir de esta información se identificaron los principales actores del sistema, entre los que se encuentran el administrador del sistema, personal de Recursos Humanos, Operaciones, Recaudo, Diésel, Monitoreo, Siniestros y Dirección. Cada uno de estos usuarios requiere acceder a funciones específicas de acuerdo con sus responsabilidades dentro de la organización.

Como resultado del análisis se definieron los principales casos de uso del sistema, tales como iniciar sesión, gestionar usuarios, registrar operadores, administrar unidades, consultar jornadas laborales, registrar suministro de combustible, generar reportes, consultar información operativa y administrar incidencias o siniestros.

Para representar gráficamente la interacción entre los usuarios y las funcionalidades principales del sistema, se elaboró un diagrama de casos de uso. Este diagrama permitió visualizar el alcance funcional de la plataforma y sirvió como base para organizar los módulos que serían desarrollados posteriormente.

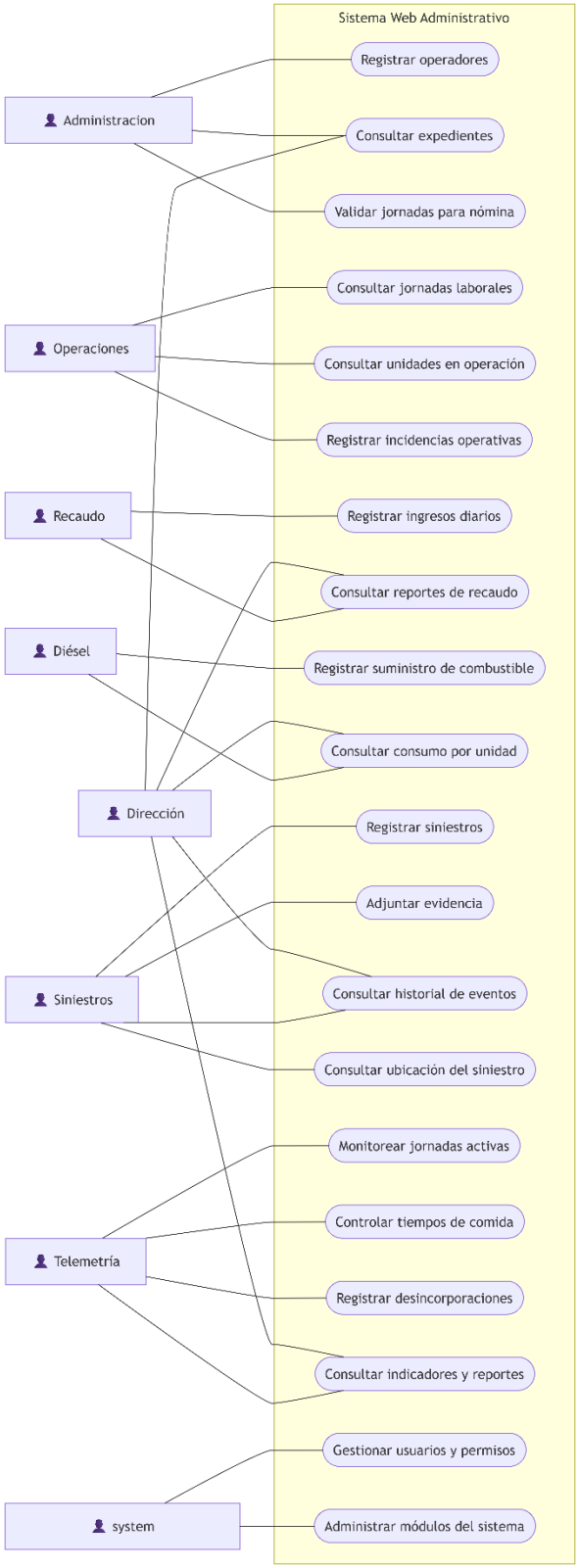

Figura 3.2 Diagrama de casos de uso del sistema administrativo

La definición de estos casos de uso permitió delimitar el alcance del sistema y establecer las funcionalidades prioritarias para cada área. De esta manera, el análisis de requerimientos sirvió como base para el diseño de la arquitectura, la estructura de la base de datos y la organización modular de la plataforma web.

3.1.2 Administración del proyecto e ingeniería de software aplicada

El desarrollo del sistema se realizó aplicando principios de ingeniería de software orientados a organizar el proceso de construcción de la plataforma, controlar el avance del proyecto y asegurar que las funcionalidades implementadas respondieran a las necesidades identificadas durante el levantamiento de requerimientos.

Para la gestión del proyecto opté por utilizar una metodología de desarrollo incremental. La elección de este enfoque estuvo directamente relacionada con las condiciones operativas de la empresa y con la necesidad de obtener resultados funcionales en periodos cortos de tiempo. Debido a que los problemas administrativos identificados afectaban diariamente la operación de la organización, no era viable esperar a que el sistema completo estuviera terminado para comenzar a utilizarlo.

Asimismo, el desarrollo, validación e implementación de la plataforma fueron realizados directamente por mí, sin contar con equipos especializados de análisis, pruebas o gestión de proyectos. Esta situación hacía poco práctico adoptar metodologías ágiles más complejas, como Scrum, las cuales requieren la participación de múltiples roles y una estructura de trabajo colaborativa. De igual manera, un modelo en cascada tradicional habría implicado completar todas las etapas de análisis, diseño, desarrollo y pruebas antes de liberar cualquier funcionalidad, retrasando significativamente la obtención de beneficios para la empresa.

Por estas razones, la metodología incremental resultó ser la alternativa más adecuada, ya que permitió desarrollar módulos funcionales de manera independiente, liberándolos progresivamente conforme eran terminados y validados. De esta forma, cada área comenzó a utilizar nuevas herramientas sin necesidad de esperar a la finalización total del sistema.

La definición de los incrementos se realizó considerando el nivel de dependencia existente entre los procesos. El primer módulo desarrollado correspondió al registro de operadores, debido a que los operadores constituyen el elemento central de la mayoría de los procesos administrativos y operativos de la organización. Posteriormente se desarrolló el módulo de administración de unidades, ya que las jornadas laborales requieren asociar cada operador con una unidad específica.

Una vez implementados estos elementos fundamentales, se desarrolló el módulo de recaudo, permitiendo asociar los ingresos generados por cada unidad con los operadores que participaron en la operación correspondiente. Posteriormente se implementó el módulo de siniestros, debido a la necesidad de documentar eventos relacionados con operadores y unidades, generando un historial digital que facilitara su consulta y seguimiento.

El siguiente incremento correspondió al módulo de mantenimiento, cuyo objetivo fue registrar y conservar el historial de reparaciones e intervenciones realizadas a las unidades. De forma

complementaria, se desarrolló el módulo de control de combustible, permitiendo asociar los suministros de diésel con cada unidad y generar información para el análisis del consumo.

Posteriormente se incorporaron funcionalidades relacionadas con la gestión de reparaciones de radios y sistemas de videovigilancia instalados en las unidades. Finalmente, se desarrollaron herramientas orientadas al área de monitoreo y telemetría, incluyendo el control de tiempos de jornada, cálculo de horas pendientes, registro de comidas solicitadas, control de siniestros y administración de desincorporaciones de unidades.

Después de cada incremento se realizaron pruebas funcionales para verificar el correcto funcionamiento de las nuevas características antes de su liberación al entorno de producción. Esta estrategia permitió detectar y corregir errores de manera temprana, además de incorporar ajustes derivados de la retroalimentación recibida por los usuarios durante la utilización diaria del sistema.

La aplicación de la metodología incremental permitió responder de forma rápida a las necesidades de la empresa, mantener un desarrollo ordenado y garantizar que cada módulo aportara beneficios inmediatos a las áreas involucradas, contribuyendo progresivamente a la centralización de la información y a la mejora de los procesos administrativos.

3.1.3 Módulos identificados y alcance funcional del sistema

La arquitectura funcional del sistema fue diseñada considerando las dependencias existentes entre los diferentes procesos administrativos de la organización. Debido a que gran parte de la información utilizada por las distintas áreas se encuentra relacionada entre sí, los módulos fueron desarrollados bajo un esquema centralizado en el que cada componente comparte una base de datos común y utiliza reglas de negocio que garantizan la integridad de la información.

El sistema se encuentra estructurado en módulos funcionales interconectados. La base de la plataforma está conformada por los módulos de autenticación, administración de usuarios, operadores y unidades, ya que estos proporcionan la información necesaria para el funcionamiento de los demás procesos administrativos.

El módulo de autenticación y control de acceso constituye el punto de entrada de la plataforma. Su función principal consiste en validar las credenciales proporcionadas por el usuario y determinar los permisos asociados a su cuenta. Para ello, la interfaz web captura la información ingresada por el usuario y realiza validaciones básicas mediante JavaScript. Posteriormente, estos datos son enviados al servidor mediante peticiones HTTP, donde son procesados por componentes desarrollados en PHP. Una vez validada la identidad del usuario, el sistema consulta los roles y permisos asignados en la base de datos y devuelve esta información al cliente. Con base en dichos permisos, la interfaz habilita únicamente los módulos autorizados para cada usuario, ocultando aquellas funcionalidades que no corresponden a su área de responsabilidad.

La utilización de roles responde a la necesidad de controlar el acceso a la información entre los distintos departamentos de la organización. Debido a que áreas como Recursos Humanos, Recaudo, Operaciones, Siniestros, Monitoreo y Dirección administran información de naturaleza distinta, fue

necesario implementar mecanismos que permitieran restringir la visualización y modificación de datos según las funciones asignadas a cada usuario.

El módulo de operadores constituye uno de los componentes principales de la plataforma debido a que gran parte de los procesos administrativos dependen de la información del personal operativo. Este módulo almacena datos personales, documentación laboral, evidencias fotográficas y demás información necesaria para identificar a cada operador dentro del sistema.

Por su parte, el módulo de unidades administra la información relacionada con los vehículos de la organización. La relación entre operadores y unidades constituye una de las principales reglas de negocio implementadas, ya que diversos procesos requieren asociar ambas entidades para garantizar la consistencia de los registros.

A partir de estos módulos base se construyen procesos más complejos. El módulo de jornadas laborales establece la relación entre operadores, unidades y actividades realizadas durante un periodo determinado. Esta información es utilizada posteriormente por otros componentes para generar indicadores operativos y administrativos.

Los módulos de recaudo, control de combustible y siniestros dependen directamente de la existencia de registros válidos de operadores, unidades y jornadas laborales. En estos casos, las reglas de negocio implementadas obligan a relacionar cada registro con las entidades correspondientes, permitiendo mantener la trazabilidad de la información y facilitar su análisis posterior.

De manera complementaria, los módulos de mantenimiento, reparación de radios, sistemas de videovigilancia y telemetría fueron diseñados para generar historiales asociados a las unidades. Esto permite conservar evidencia de las intervenciones realizadas y mantener un seguimiento detallado del estado operativo de cada vehículo.

Finalmente, la información generada por todos los módulos es utilizada por el subsistema de reportes y analítica. Este componente consolida los datos almacenados en la base de datos y genera indicadores, consultas especializadas, estadísticas y visualizaciones que apoyan las actividades de supervisión y toma de decisiones dentro de la organización.

La integración de estos módulos permitió construir una plataforma centralizada en la que la información fluye de manera estructurada entre los distintos procesos administrativos, garantizando la consistencia de los datos y facilitando la administración de la operación mediante una arquitectura unificada.

3.1.4 Desarrollo incremental y tecnologías utilizadas por etapa

El desarrollo del sistema se realizó mediante una estrategia incremental, permitiendo incorporar nuevas funcionalidades de manera progresiva conforme las necesidades de la organización lo requerían. Cada incremento fue construido, probado e implementado antes de iniciar el siguiente, permitiendo que las áreas comenzaran a utilizar nuevas herramientas sin esperar la finalización completa del proyecto.

Primer incremento: Dashboard con módulos para las diferentes áreas

El primer componente desarrollado correspondió al dashboard principal del sistema y al mecanismo de autenticación y control de acceso basado en roles. Este incremento tuvo como objetivo establecer la estructura inicial de la plataforma y permitir que los usuarios visualizaran únicamente los módulos correspondientes a sus funciones dentro de la organización.

Para ello se desarrolló una interfaz principal desde la cual se presentan los distintos módulos disponibles dentro del sistema. La visualización de cada módulo depende de los permisos asignados al usuario durante el proceso de autenticación. Cuando un usuario inicia sesión, el sistema consulta los roles asociados a su cuenta y habilita únicamente las opciones autorizadas dentro del menú principal.

La implementación de este mecanismo permitió validar el funcionamiento del modelo de seguridad basado en roles antes de comenzar el desarrollo de los módulos operativos. Inicialmente se implementó el rol **system**, el cual cuenta con acceso completo a todas las funcionalidades de la plataforma y fue utilizado para tareas de desarrollo, administración y pruebas.

La lógica de autenticación y autorización fue desarrollada en PHP, mientras que la interacción de la interfaz se implementó mediante HTML5, CSS3 y JavaScript. La información de usuarios, roles y permisos se almacena en PostgreSQL y es consultada durante cada inicio de sesión para determinar las funcionalidades disponibles para cada usuario.

Segundo incremento: Administración de operadores

El segundo módulo desarrollado correspondió al registro y administración de operadores, debido a que esta entidad constituye la base de gran parte de los procesos administrativos y operativos del sistema. Para su implementación se utilizó HTML5 en la construcción de formularios de captura, aprovechando sus capacidades nativas de validación y compatibilidad con tecnologías web modernas.

La presentación visual fue desarrollada mediante CSS3, permitiendo construir interfaces organizadas y fáciles de utilizar por el personal administrativo. La interacción del lado del cliente fue implementada mediante JavaScript, encargado de validar datos antes de su envío, gestionar eventos de la interfaz y realizar solicitudes asíncronas al servidor.

La lógica de negocio fue desarrollada en PHP, donde se implementaron las validaciones del lado del servidor, el procesamiento de formularios y la comunicación con PostgreSQL mediante PDO utilizando sentencias preparadas. La información principal de este módulo se almacena en la tabla **empleado_chofer**, la cual contiene datos personales, laborales, biométricos y documentales de los operadores.

Durante esta etapa se incorporó el rol **administracion**, permitiendo al personal de Recursos Humanos y Administración gestionar la información de los operadores, documentación laboral y registros relacionados con el personal operativo.

Tercer incremento: Administración de unidades

Una vez implementado el registro de operadores, se desarrolló el módulo de administración de unidades. Este módulo permite almacenar la información técnica y administrativa de los autobuses utilizados dentro de la operación del Corredor.

La información se centraliza en la tabla **autobus**, donde se registran datos como número económico, placas, número de serie, número de motor, vigencia del seguro, aseguradora, kilometraje, capacidad del tanque de combustible y estatus operativo de la unidad.

La incorporación de este módulo permitió contar con un catálogo centralizado de unidades, necesario para relacionar posteriormente las jornadas laborales, los registros de recaudo, los suministros de combustible, los mantenimientos y los siniestros.

Durante esta etapa se incorporó el rol **operaciones**, debido a que esta área es la encargada de supervisar la disponibilidad y utilización de las unidades durante la operación diaria. Este rol permite consultar información relacionada con los autobuses y gestionar las actividades operativas asociadas a las unidades registradas dentro del sistema.

Cuarto incremento: Gestión de jornadas laborales

El siguiente incremento correspondió a la gestión de jornadas laborales, debido a que representa el punto de integración entre operadores y unidades. Este módulo sustituyó el uso de hojas físicas utilizadas para registrar entradas, salidas y horas laboradas por los operadores.

La información se almacena en la tabla **historial_jornada_completa**, la cual registra fechas de entrada y salida, horarios, unidad asignada, kilometrajes, niveles de combustible, tiempos de comida, observaciones operativas y duración total de la jornada.

Como regla de negocio principal, una jornada únicamente puede registrarse cuando existe una asociación válida entre un operador y una unidad previamente registrados en el sistema. Esto permitió mantener la consistencia de la información y facilitar la trazabilidad de las actividades realizadas durante cada turno.

Durante esta etapa se incorporó el rol **checadores**, debido a que este personal es responsable de supervisar y registrar las jornadas laborales de los operadores. Este rol cuenta con acceso a las funcionalidades relacionadas con el control de asistencia, asignación de unidades, seguimiento de horarios y consulta de información operativa relacionada con las jornadas activas y finalizadas.

Quinto incremento: Recaudo

Posteriormente se desarrolló el módulo de recaudo, permitiendo registrar los ingresos generados por las unidades durante la operación diaria.

La información es almacenada en la tabla **historial_recaudo**, donde cada registro se relaciona con una unidad y uno o dos operadores responsables de la jornada correspondiente. Esta relación permite mantener trazabilidad entre la operación realizada y los ingresos obtenidos.

Con la implementación de este módulo se creó el rol **recaudo**, el cual fue configurado con acceso a los módulos de Recaudo, Préstamos, Mensualidades y Documentos. Esto permitió que el personal

encargado del conteo y control de ingresos pudiera registrar la información sin acceder a otros módulos de la plataforma.

Sexto incremento: Gestión de siniestros

El módulo de siniestros fue incorporado para generar expedientes digitales asociados a eventos ocurridos durante la operación.

La información principal se almacena en la tabla **siniestros**, la cual registra la unidad involucrada, operador responsable, evidencias fotográficas, ubicación geográfica, aseguradora, deducibles, pagos relacionados y observaciones generales. Este módulo permitió centralizar información que anteriormente se encontraba dispersa en dispositivos móviles y registros físicos.

Durante esta etapa se creó el rol **siniestros**, permitiendo que únicamente el personal responsable pudiera registrar, consultar y dar seguimiento a los accidentes e incidentes operativos registrados dentro del sistema.

Séptimo incremento: Control de mantenimiento

Posteriormente se desarrolló el módulo de mantenimiento con el objetivo de generar historiales de intervenciones realizadas a las unidades.

La información se almacena principalmente en la tabla **historial_electro_mecanica**, donde se registran reparaciones, diagnósticos, acciones correctivas, refacciones utilizadas y personal responsable de las intervenciones.

Con este incremento se incorporaron los roles **electro_mecanica**, **almacen**, **hojalateria** y **golpes**, permitiendo segmentar las actividades relacionadas con mantenimiento, reparaciones, control de inventarios y seguimiento de daños físicos de las unidades de acuerdo con las responsabilidades de cada departamento.

Octavo incremento: Control de combustible

El módulo de diésel fue incorporado para registrar los suministros de combustible realizados a las unidades.

La información principal se almacena en la tabla **historial_diesel**, permitiendo relacionar cada carga con una unidad específica y generar indicadores de consumo, rendimiento y seguimiento operativo.

Durante esta etapa se creó el rol **diesel**, con acceso exclusivo a las funcionalidades relacionadas con el registro y análisis del combustible suministrado a las unidades.

Noveno incremento: Telemetría y analítica operativa

Finalmente se desarrollaron los módulos orientados al monitoreo operativo, análisis de información y supervisión en tiempo real. Estos componentes procesan la información generada por los módulos anteriores para calcular tiempos de jornada, horas faltantes, periodos de comida, incidencias operativas, desincorporaciones y diversos indicadores utilizados para la supervisión administrativa.

Durante esta etapa se incorporó el rol **tele**, el cual integra acceso a los módulos de Telemetría, Radios, Cámaras, Analítica Diésel, Letreros y Documentos. Asimismo, se desarrollaron herramientas de monitoreo en tiempo real para la supervisión de jornadas activas, registro de incidencias operativas y seguimiento de eventos relevantes dentro de la operación.

Posteriormente se incorporó el rol **direccion**, destinado a los usuarios responsables de la supervisión administrativa y consulta de indicadores estratégicos para la toma de decisiones.

Finalmente, conforme surgieron nuevas necesidades operativas, se añadió el rol **edicion**, utilizado para tareas específicas relacionadas con la actualización y consulta controlada de información dentro de la plataforma.

Integración tecnológica de la solución

La arquitectura implementada se basa en una comunicación cliente-servidor de tres capas. La capa de presentación está compuesta por HTML5, CSS3 y JavaScript. La capa de negocio fue desarrollada mediante PHP y la capa de persistencia utiliza PostgreSQL como sistema gestor de bases de datos.

El flujo de información inicia cuando el usuario captura datos en la interfaz web. JavaScript realiza validaciones preliminares y envía la información al servidor mediante peticiones HTTP. Posteriormente PHP procesa la solicitud, aplica las reglas de negocio correspondientes y ejecuta sentencias SQL mediante PDO. Finalmente PostgreSQL almacena o recupera la información solicitada y la respuesta es enviada nuevamente al cliente en formato JSON para su visualización.

Pruebas realizadas

Debido a que el desarrollo, validación e implementación del sistema fueron realizados directamente por mí, las pruebas funcionales se ejecutaron de manera manual durante cada incremento. No se utilizaron frameworks especializados para pruebas automatizadas ni herramientas de integración continua.

Las pruebas realizadas corresponden principalmente a pruebas funcionales de caja negra, verificando que los formularios capturaran correctamente la información, que las validaciones impidieran registros inválidos, que las consultas devolvieran los resultados esperados y que la información fuera almacenada correctamente en PostgreSQL.

Asimismo, se realizaron pruebas de integración para verificar la comunicación entre los módulos desarrollados y la base de datos, validando operaciones de creación, consulta y actualización de registros.

Seguridad e integridad de la información

Como medida de protección, el sistema implementa autenticación mediante usuario y contraseña, control de sesiones, gestión de roles y permisos, y restricciones de acceso según el departamento al que pertenece cada usuario.

La comunicación con PostgreSQL se realiza mediante PDO utilizando sentencias preparadas, reduciendo el riesgo de ataques de inyección SQL. Adicionalmente, las operaciones sobre la

información fueron diseñadas bajo un esquema CRUD parcial, permitiendo únicamente operaciones de creación, consulta y actualización. No se implementaron operaciones de eliminación física de registros, con el objetivo de preservar la trazabilidad histórica de la información y mantener la integridad de los datos almacenados.

3.1.5 Cronograma de actividades

La planificación del proyecto se realizó considerando las necesidades operativas de cada departamento y el enfoque incremental adoptado durante el desarrollo. Debido a que la organización requería soluciones funcionales en el menor tiempo posible, las actividades fueron priorizadas de acuerdo con la importancia de cada área dentro de la operación general de la empresa.

El cronograma se estructuró considerando un periodo aproximado de doce meses. Durante las primeras etapas se realizaron las actividades de análisis de requerimientos, diseño de la arquitectura y construcción de la base de datos. Posteriormente se inició el desarrollo progresivo de los módulos administrativos siguiendo el orden de prioridad establecido durante el levantamiento de requerimientos.

El primer departamento atendido fue Recursos Humanos, mediante la implementación del módulo de operadores. Este componente constituyó la base de la plataforma debido a que la mayoría de los procesos posteriores requieren relacionar información con los operadores registrados.

Posteriormente se desarrolló el módulo de administración de unidades, permitiendo establecer las relaciones necesarias entre operadores y vehículos para la gestión de jornadas laborales y procesos operativos posteriores.

Una vez consolidados estos componentes principales, se continuó con el desarrollo de los módulos correspondientes a Recaudo, Siniestros, Mantenimiento, Control de Diésel, Reparación de Radios y Sistemas de Videovigilancia, finalizando con las herramientas de Monitoreo, Telemetría y Analítica Operativa.

Durante todo el proceso se realizaron pruebas funcionales, correcciones y ajustes derivados de la retroalimentación obtenida por los usuarios de cada departamento. Finalmente, se efectuó la integración completa del sistema, la elaboración de manuales de usuario y la documentación técnica correspondiente.

Actividad / Departamento	Mes 1-2	Mes 3-4	Mes 5-6	Mes 7-8	Mes 9-10	Mes 11-12
Levantamiento de requerimientos y análisis	[redacted]	[redacted]				
Diseño de base de datos y arquitectura	[redacted]	[redacted]				
Infraestructura y configuración de servidores	[redacted]	[redacted]				
Recursos Humanos (Registro de operadores)		[redacted]	[redacted]			

Administración de unidades		[redacted]	[redacted]			
Jornadas laborales y operaciones			[redacted]	[redacted]		
Recaudo			[redacted]	[redacted]		
Siniestros			[redacted]	[redacted]		
Mantenimiento electro-mecánico				[redacted]	[redacted]	
Control de diésel				[redacted]	[redacted]	
Radios y videovigilancia					[redacted]	[redacted]
Monitoreo y telemetría					[redacted]	[redacted]
Reportes y analítica operativa					[redacted]	[redacted]
Pruebas funcionales y validación		[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Implementación y ajustes en producción				[redacted]	[redacted]	[redacted]
Manuales de usuario					[redacted]	[redacted]
Documentación técnica e informe final					[redacted]	[redacted]

Tabla 3.1 Cronograma general del proyecto.

La organización de las actividades por departamentos permitió atender de manera prioritaria las necesidades más críticas de la empresa, garantizando que cada módulo aportara beneficios operativos desde etapas tempranas del proyecto y facilitando la adopción progresiva de la plataforma por parte de los usuarios.

3.2 Arquitectura general del sistema

La arquitectura del sistema web administrativo fue diseñada con el objetivo de centralizar la gestión de información de las diferentes áreas operativas y administrativas de la empresa. La plataforma funciona mediante una aplicación web accesible desde navegadores, lo que permite que los usuarios autorizados puedan consultar, registrar y actualizar información desde los equipos disponibles dentro de la organización.

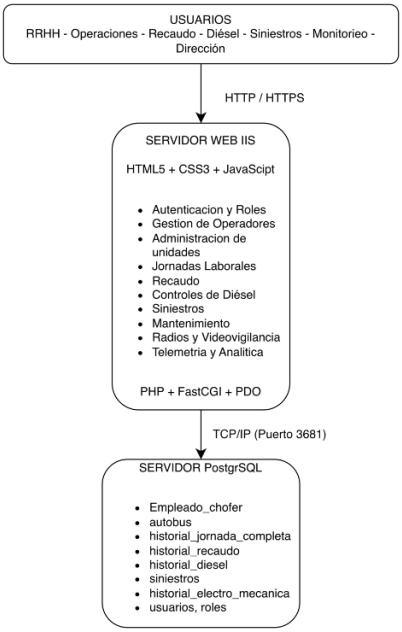

Figura 3.3 Arquitectura general e integracion de modulos del sistema

El sistema se estructura bajo una arquitectura cliente-servidor organizada en tres capas principales: presentación, lógica de negocio y persistencia de datos. Esta estructura se encuentra alineada con el enfoque incremental utilizado durante el desarrollo, donde cada nuevo módulo fue incorporado sobre una base previamente validada y operativa.

La capa de presentación está compuesta por HTML5, CSS3 y JavaScript. A través de esta capa se capturan los datos introducidos por el usuario y se realizan validaciones preliminares antes de ser enviados al servidor.

Cuando un usuario interactúa con algún módulo del sistema, JavaScript envía la información mediante solicitudes HTTP o HTTPS hacia el servidor web IIS. Una vez recibida la petición, el servidor ejecuta los archivos PHP mediante FastCGI, donde se aplican las reglas de negocio, validaciones adicionales y controles de acceso definidos para cada proceso.

La lógica de negocio implementada en PHP fue desarrollada considerando las dependencias entre módulos identificadas durante el análisis de requerimientos. Por esta razón, los módulos de operadores y unidades constituyen la base funcional de la plataforma, ya que procesos posteriores como jornadas laborales, recaudo, control de combustible y siniestros dependen directamente de la existencia de dichos registros.

Para el acceso a la información almacenada, PHP utiliza PDO (PHP Data Objects) con sentencias preparadas, estableciendo una conexión TCP/IP con PostgreSQL. Este mecanismo permite ejecutar consultas, inserciones y actualizaciones de forma segura, reduciendo el riesgo de inyección SQL y garantizando la integridad de los datos.

La capa de persistencia está conformada por PostgreSQL, donde se almacena la información generada por todos los módulos del sistema. Las tablas principales fueron diseñadas de acuerdo con los incrementos desarrollados durante el proyecto, permitiendo mantener relaciones entre operadores, unidades, jornadas laborales, recaudo, combustible, siniestros y mantenimiento.

Asimismo, la arquitectura incorpora un mecanismo de autenticación y autorización basado en roles. Cuando un usuario inicia sesión, el sistema consulta los permisos asociados a su cuenta y habilita únicamente los módulos autorizados para su área de trabajo. Esta estrategia permite mantener la separación de responsabilidades entre departamentos y restringir el acceso a información sensible según las funciones de cada usuario.

Gracias a esta arquitectura, la plataforma logró integrar en un único entorno los diferentes procesos administrativos de la organización, permitiendo que la información fluya entre módulos relacionados y manteniendo un repositorio centralizado para la gestión de los datos.

3.2.1 Control de acceso y roles del sistema

El sistema web implementa un mecanismo de control de acceso basado en roles almacenados en la base de datos. Este modelo permite determinar qué módulos y funcionalidades pueden ser utilizados por cada usuario de acuerdo con sus responsabilidades dentro de la organización.

La definición de roles formó parte de la etapa de análisis y diseño del sistema, donde se identificó la necesidad de restringir el acceso a la información entre los distintos departamentos de la empresa. Debido a que la plataforma integra procesos administrativos, operativos, de mantenimiento, recaudo, telemetría y siniestros, se estableció un modelo de control de acceso basado en roles que permitiera segmentar la información y garantizar que cada área únicamente tuviera acceso a los módulos y datos relacionados con sus funciones dentro de la organización.

Cada usuario registrado en la plataforma tiene asociados uno o varios roles que representan las áreas o funciones a las que pertenece. Cuando un usuario inicia sesión, el sistema valida sus credenciales y consulta los roles asignados a su cuenta. Posteriormente, la aplicación habilita dinámicamente los módulos permitidos dentro de la interfaz y oculta aquellos para los cuales no posee autorización.

Este mecanismo se implementa mediante un mapeo entre los roles definidos en el sistema y las categorías del menú principal. De esta manera, la interfaz se adapta automáticamente al perfil del usuario, permitiendo visualizar únicamente las secciones que corresponden a su área de trabajo. Esta estrategia facilita el uso de la plataforma y reduce el riesgo de accesos no autorizados a información sensible.

La administración de permisos se realiza directamente desde la base de datos, permitiendo asignar o retirar accesos sin necesidad de modificar el código fuente de la aplicación. Gracias a este enfoque, el sistema puede adaptarse fácilmente a cambios organizacionales o a la incorporación de nuevos módulos y departamentos.

Adicionalmente, el control de acceso se complementa con el uso de sesiones autenticadas, validaciones de seguridad en el servidor y restricciones implementadas en cada módulo, evitando que un usuario pueda acceder directamente a funcionalidades para las cuales no posee permisos, aun cuando conozca la dirección URL correspondiente.

Dentro de los módulos habilitados, los usuarios pueden realizar operaciones relacionadas con el registro, consulta y actualización de información. Sin embargo, el sistema fue diseñado bajo una política de conservación de datos, por lo que no se permite la eliminación física de registros. Esta

decisión busca preservar el historial de información, mantener la trazabilidad de los procesos administrativos y facilitar futuras auditorías o consultas históricas.

La Tabla 3.2 presenta la matriz de control de acceso implementada en la plataforma, mostrando la relación entre los roles definidos y los módulos disponibles para cada uno de ellos.

Rol \ Módulo	Admon	Operación	Checador	Recaudo	Siniestros	Mecánica	Diésel	Telemetría	Dirección	Documentación
administracion	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓
operaciones	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
checadores	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓
recaudo	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
siniestros	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
electro_mecanica	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓
diesel	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
tele	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
direccion	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
system	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabla 3.2 Roles del sistema y módulos habilitados.

Como puede observarse, la mayoría de los roles poseen acceso únicamente a los módulos necesarios para el desempeño de sus funciones. El único perfil con acceso global es el rol **system**, utilizado para tareas de administración, configuración, soporte y mantenimiento de la plataforma. Esta estructura permitió mantener la segregación de funciones entre departamentos y reducir el riesgo de acceso no autorizado a la información administrativa y operativa del Corredor.

3.2.2 Menú del sistema con módulos habilitados según el rol del usuario

Después de que un usuario inicia sesión en la plataforma, el sistema muestra el menú principal de navegación, el cual permite acceder a los diferentes módulos disponibles dentro del sistema web administrativo. Las categorías y opciones visibles dentro del menú dependen directamente de los roles asignados al usuario en la base de datos.

El proceso inicia desde la interfaz de inicio de sesión, donde el usuario ingresa su nombre de usuario, contraseña. Mediante JavaScript se validan los campos obligatorios y posteriormente se envían los datos al archivo PHP encargado de procesar la autenticación. Este envío se realiza mediante una petición HTTP de tipo POST.

```
const response = await fetch("../backend/Archivo_accesar.php", {
    method: "POST",
    body: formData,
    credentials: "include"
});
```

```
const data = await response.json();
```

En el servidor, PHP recibe las credenciales del usuario, realiza la conexión con PostgreSQL y valida que los datos ingresados correspondan a un usuario registrado y activo. Una vez validado el acceso, el sistema consulta los roles y permisos asociados al usuario mediante su identificador.

```
$rolesStmt = $pdo->prepare("
    SELECT r.clave
    FROM modulos_activos_usuarios ur
    INNER JOIN roles_app r ON r.id_rol = ur.id_rol
    WHERE ur.id_usuario = :id_usuario
      AND ur.activo = TRUE
      AND r.activo = TRUE
");
$rolesStmt->execute([':id_usuario' => $user['id_usuario']]);
$roles = $rolesStmt->fetchAll(PDO::FETCH_COLUMN);
```

Posteriormente, PHP devuelve al frontend una respuesta en formato JSON que contiene los datos del usuario, sus roles y permisos. Esta respuesta permite que JavaScript pueda determinar qué módulos deben mostrarse en el dashboard.

```
jsonResponse([
    'success' => true,
    'message' => 'Login exitoso.',
    'user' => [
        'id_usuario' => (int)$user['id_usuario'],
        'username' => $user['username'],
        'nombre' => $user['nombre'],
        'email' => $user['email'] ?? '',
        'dbname' => $dbname,
        'roles' => $roles,
        'permisos' => $permisos
    ]
]);
```

Cuando la respuesta es recibida correctamente por JavaScript, los roles y permisos se almacenan temporalmente en el navegador mediante localStorage. Esto permite conservar la información del usuario durante la sesión activa y utilizarla para personalizar la interfaz.

```
localStorage.setItem("username", data.user.username);
localStorage.setItem("roles", JSON.stringify(data.user.roles));
localStorage.setItem("permisos", JSON.stringify(data.user.permisos));
```

Al cargar el dashboard, el sistema verifica que exista una sesión válida. Para ello, el archivo auth.js realiza una petición al servidor y, si la sesión es correcta, recupera nuevamente la información del usuario, incluyendo sus roles y permisos. En caso de que la sesión no sea válida, el sistema redirige automáticamente al usuario a la pantalla de inicio de sesión.

```
async function verifySessionOrRedirect() {
  try {
    const user = await fetchMySession();

    localStorage.setItem("username", user.username || "");
    localStorage.setItem("dbname", user.dbname || "");
    localStorage.setItem("roles", JSON.stringify(user.roles || []));
    localStorage.setItem("permisos", JSON.stringify(user.permisos || []));

    return user;
  } catch (error) {
    console.error("Sesión inválida:", error);
    window.location.href = "entrada.html";
    return null;
  }
}
```

Una vez validada la sesión, el dashboard ejecuta la función encargada de habilitar u ocultar los módulos del sistema de acuerdo con los roles asignados al usuario.

```
updateButtonsVisibility(user.roles || [], user.username || '');
```

La función **updateButtonsVisibility()** contiene un mapeo entre cada rol del sistema y los módulos que puede visualizar. Por ejemplo, el rol administracion habilita los módulos de Administración, Recursos Humanos y Documentos; el rol recaudo habilita Recaudo, Préstamos, Mensualidades y Documentos; mientras que el rol system habilita todos los módulos disponibles.

```
const roleMenuMap = {
  administracion: [
    { category: "admin-category", item: "admin-menu" },
    { category: "rh-category", item: "rh-menu" },
    { category: "documentos-category", item: "documentos-menu" }
  ],
  recaudo: [
    { category: "recaudo-category", item: "recaudo-menu" },
    { category: "prestamos-category", item: "prestamos-menu" },
    { category: "mensualidades-category", item: "mensualidades-menu" },
    { category: "documentos-category", item: "documentos-menu" }
  ],
  system: [
    { category: "admin-category", item: "admin-menu" },
```

```
        { category: "rh-category", item: "rh-menu" },
        { category: "siniestros-category", item: "siniestros-menu" },
        { category: "checadores-category", item: "checadores-menu" },
        { category: "recaudo-category", item: "recaudo-menu" },
        { category: "mecanica-category", item: "mecanica-menu" },
        { category: "almacen-category", item: "almacen-menu" },
        { category: "operaciones-category", item: "operaciones-menu" },
        { category: "diesel-category", item: "diesel-menu" },
        { category: "telemetria-category", item: "telemetria-menu" },
        { category: "direccion-category", item: "direccion-menu" },
        { category: "documentos-category", item: "documentos-menu" }
    ]
};
```

Antes de mostrar los módulos autorizados, el sistema oculta todos los elementos del menú. Posteriormente recorre los roles del usuario, identifica los módulos permitidos y cambia su propiedad visual para mostrarlos dentro del dashboard.

```
document.querySelectorAll('.menu-category, .menu-item').forEach(element => {
    element.style.display = 'none';
});
const visibleItems = new Set();
user_roles.forEach(role => {
    const items = roleMenuMap[role] || [];
    items.forEach(menu => {
        visibleItems.add(menu.category);
        visibleItems.add(menu.item);
    });
});
visibleItems.forEach(id => {
    const element = document.getElementById(id);
    if (element) element.style.display = 'block';
});
```

De esta manera, el menú del sistema se construye dinámicamente de acuerdo con los permisos del usuario. Si un usuario no cuenta con un rol determinado, los módulos asociados permanecen ocultos y no forman parte de su navegación principal.

Este mecanismo permite mantener una separación funcional entre las diferentes áreas de la empresa, evitando que usuarios de un departamento visualicen módulos que no corresponden a sus actividades. Además, facilita la administración del sistema, ya que la habilitación de módulos depende de los roles asignados en la base de datos y no de configuraciones manuales en cada equipo cliente.

En las siguientes figuras se presentan ejemplos del menú del sistema mostrado a distintos usuarios, permitiendo observar cómo varían los módulos disponibles dependiendo del rol asignado.

Módulo de administración del Personal y RRHH

Figura 3.4 Menú principal del sistema para usuario con rol de administración.

Módulo de Supervisores de Operación

Figura 3.5 Menú principal del sistema para usuario con rol checadores y golpes

Módulo de Siniestros

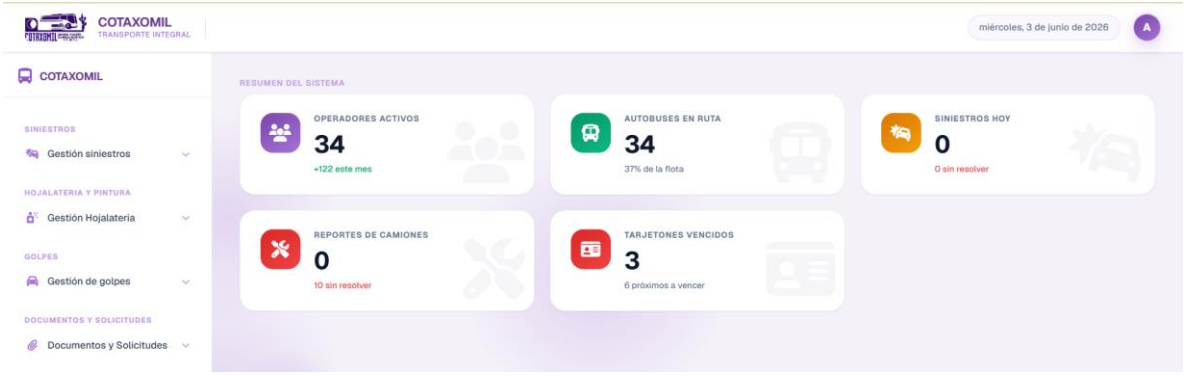

Figura 3.6 Menú principal del sistema para usuario con rol siniestros

Módulo Diesel

Figura 3.7 Menú principal del sistema para usuario con rol Diesel

Módulo Recaudo

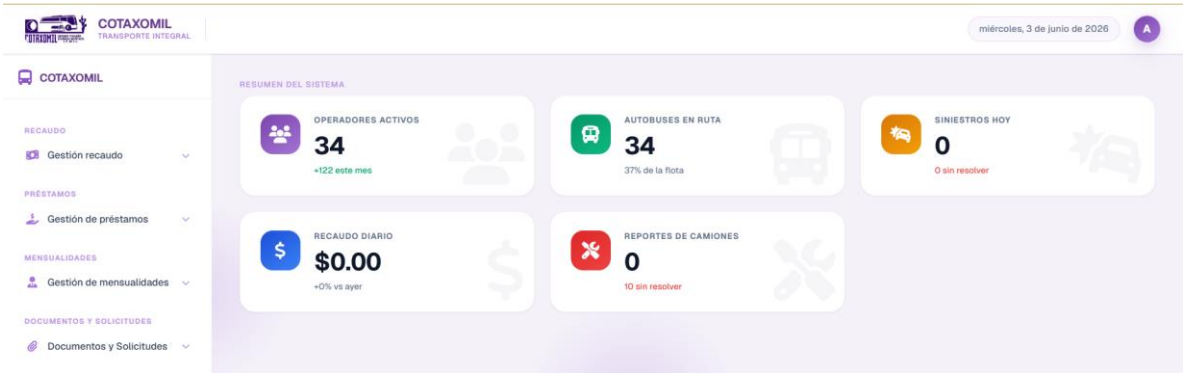

Figura 3.8 Menú principal del sistema para usuario con rol Recaudo

Módulo Mantenimiento

Figura 3.9 Menú principal del sistema para usuario con rol Electro-mecanica

Modulo Dirección

Figura 3.10 Menú principal del sistema para usuario con rol dirección

Modulo Telemetria

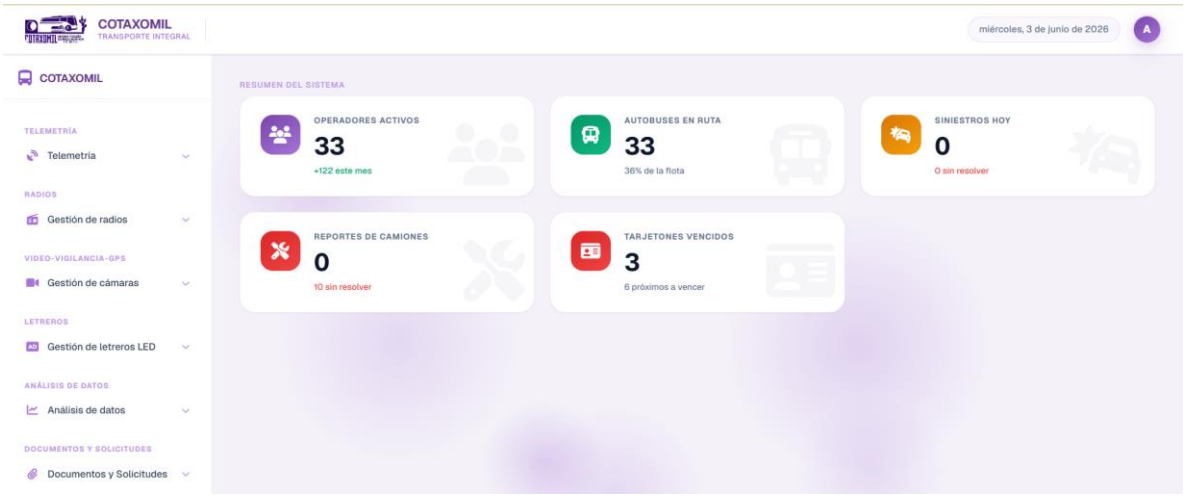

Figura 3.11 Menú principal del sistema para usuario con rol tele

Modulo System

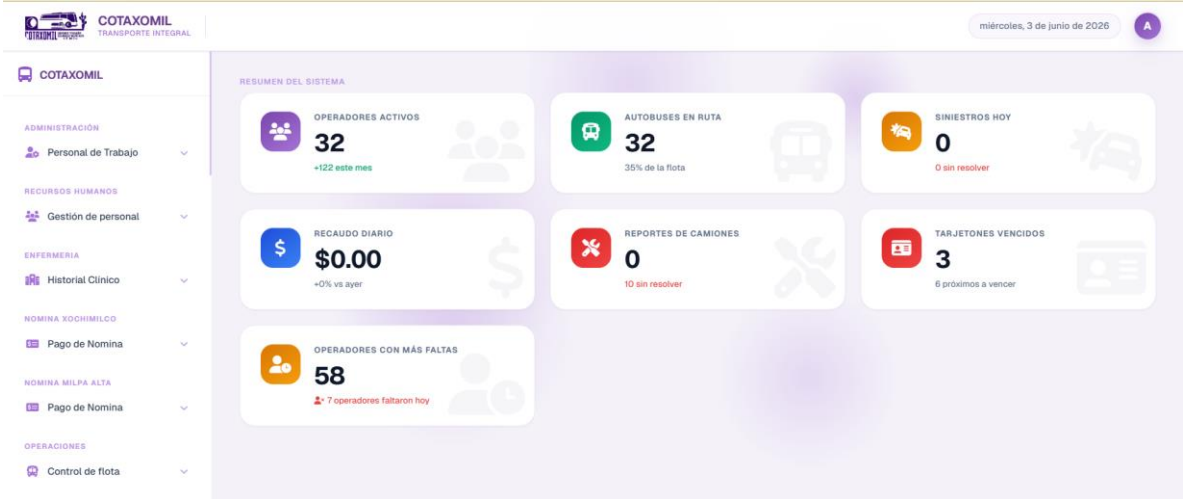

Figura 3.12 Menú principal del sistema para usuario con rol system

3.3 Front-End

El *front-end* del sistema fue diseñado con el objetivo de proporcionar una interfaz clara, intuitiva y accesible para el personal administrativo de la empresa. Debido a que los usuarios finales no necesariamente cuentan con formación técnica en sistemas de información, se buscó desarrollar una interfaz sencilla que permitiera realizar las tareas administrativas de manera rápida y comprensible.

Para la construcción de la interfaz se utilizaron tecnologías web estándar como **HTML, CSS y JavaScript**, las cuales permiten desarrollar aplicaciones accesibles desde un navegador web sin necesidad de instalar software adicional en los equipos de los usuarios.

El lenguaje **HTML** se utilizó para estructurar las páginas del sistema, definiendo formularios de captura, tablas de consulta, menús de navegación y los diferentes componentes que conforman la interfaz. Por su parte, **CSS** se empleó para definir el diseño visual de la plataforma, incluyendo la distribución de los elementos, los estilos de los formularios, los botones de interacción y la organización de los contenidos en pantalla.

Adicionalmente, **JavaScript** fue utilizado para mejorar la interactividad del sistema, implementando validaciones de datos del lado del cliente, previsualización de imágenes antes de su carga y el envío asíncrono de información al servidor. Estas funcionalidades permiten mejorar la experiencia del usuario al reducir tiempos de espera y evitar recargas innecesarias de la página.

La navegación dentro del sistema se organiza mediante menús y submenús que permiten acceder a los diferentes módulos administrativos de la plataforma. Asimismo, se implementaron tablas dinámicas que facilitan la consulta de información almacenada en la base de datos, permitiendo visualizar registros de operadores, unidades y jornadas laborales de forma ordenada.

En términos de diseño visual, se procuró mantener una identidad gráfica consistente mediante el uso de colores, tipografías y estilos homogéneos en toda la aplicación. Además, el sistema fue diseñado considerando principios de **diseño responsivo**, lo que permite su utilización tanto en equipos de escritorio como en dispositivos móviles, facilitando su uso en distintos entornos de trabajo dentro de la empresa.

3.3.1 Módulo de registro de operadores

Uno de los módulos principales del sistema corresponde al registro de operadores, el cual permite almacenar la información relacionada con el personal operativo del Corredor. Este módulo facilita la captura de datos personales y laborales de los operadores, así como la incorporación de evidencias fotográficas asociadas a su expediente administrativo.

El formulario de registro permite capturar información como número de empleado, nombre completo, RFC, número de seguridad social, CURP, salario base, tipo de jornada laboral, fecha de vencimiento del tarjetón, datos de contacto y fecha de ingreso. Asimismo, el sistema permite cargar fotografías relacionadas con la identificación del operador, tales como la credencial, el tarjetón y la fotografía del operador.

La interfaz de este módulo fue diseñada mediante un formulario web que organiza los campos de captura de manera clara para facilitar el registro de información por parte de los usuarios administrativos.

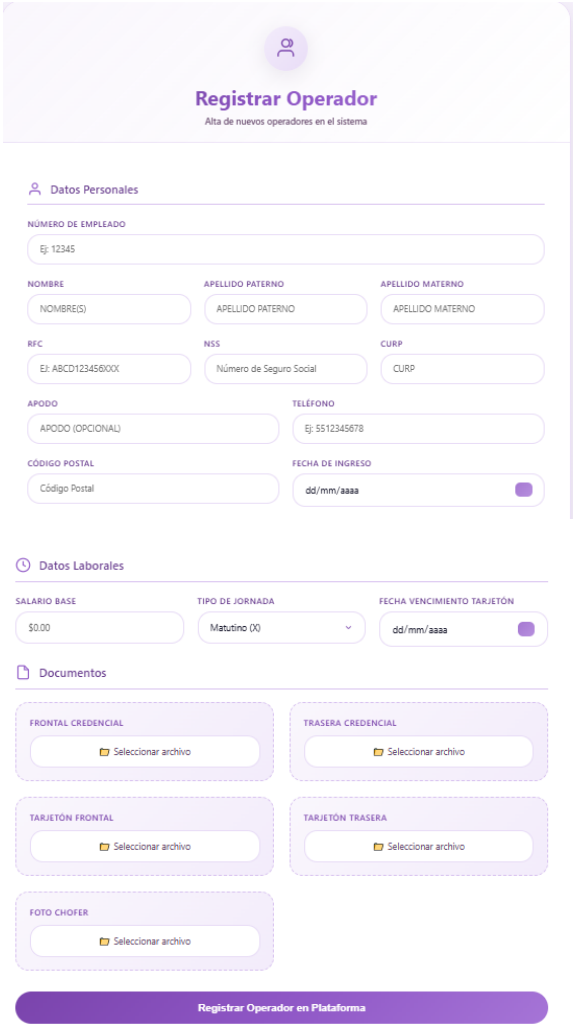

figura 3.13 Formulario de registro de operadores en el sistema web administrativo.

3.3.2 Módulo de registro de unidades (Autobuses)

El módulo de **registro de unidades** permite capturar y administrar la información relacionada con los autobuses que forman parte de la flota operativa del Corredor. Este módulo facilita el almacenamiento de datos técnicos y administrativos de cada unidad, permitiendo mantener un control actualizado del parque vehicular utilizado en la operación diaria del sistema de transporte.

Dentro de este formulario se registran datos relevantes de cada autobús, tales como el **número económico (ECO)**, las **placas de circulación**, el **número de serie**, el **número de motor**, la **vigencia del seguro**, el **nombre de la aseguradora** y el **tipo de autobús**. Esta información permite llevar un seguimiento adecuado de las unidades y facilita la consulta de datos necesarios para actividades administrativas, operativas y de mantenimiento.

La interfaz del módulo fue diseñada mediante un formulario web que organiza los campos de captura de forma clara para el usuario. Además, se incorporaron validaciones básicas para asegurar que los datos obligatorios sean registrados antes de enviar la información al servidor.

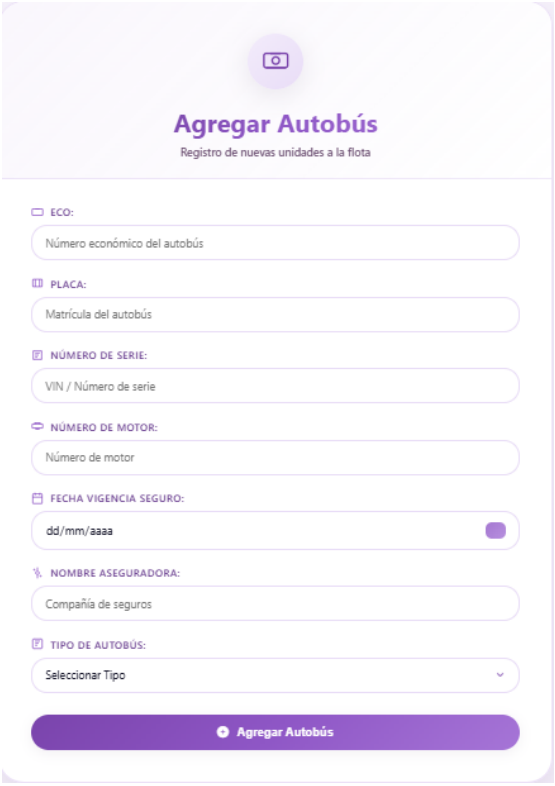

figura 3.14 Formulario de registro de unidades (autobuses) en el sistema web administrativo.

3.3.3 Módulo de gestión de jornadas laborales

El sistema incorpora un módulo para la **gestión de jornadas laborales de los operadores**, el cual permite registrar el inicio, seguimiento y finalización de las jornadas de trabajo asociadas a las unidades de transporte. Este módulo tiene como objetivo facilitar el control operativo diario del Corredor, permitiendo registrar información relevante relacionada con cada recorrido realizado por los operadores.

La gestión de jornadas se divide en tres procesos principales: **inicio de jornada, finalización de jornada y relevo de operador**. Cada uno de estos procesos permite registrar información específica sobre el estado de la unidad y del operador en distintos momentos de la operación.

3.3.3.1 Inicio de jornada

El proceso de **inicio de jornada** permite registrar el momento en que un operador comienza su turno de trabajo con una unidad específica. Durante este proceso se selecciona el operador y el autobús que utilizará, además de capturar información inicial del estado del vehículo.

Entre los datos registrados se encuentran el **kilometraje inicial de la unidad**, el **nivel de diésel**, **AdBlue**, **nivel de aceite** y el **tipo de jornada asignada**. Asimismo, el sistema permite visualizar la fotografía del operador para facilitar su identificación y evitar errores en la selección del personal.

También hay restricciones sobre la apertura de las jornadas, donde un operador no puede tener 2 jornadas activas al mismo tiempo.

Un operador no puede estar en 2 autobuses al mismo tiempo, ni un autobús puede estar activo con 2 operadores.

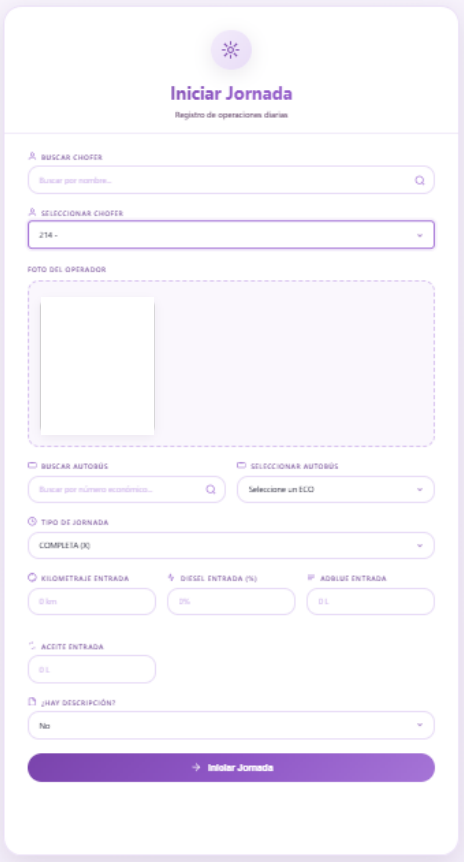

Figura 3.15 Inicio de jornada de operador en el sistema web administrativo.

3.3.3.2 Finalización de jornada

El módulo también permite registrar la **finalización de la jornada**, proceso en el cual se capturan los datos finales de operación del autobús una vez concluido el turno del operador.

Durante esta etapa se registran valores como el **kilometraje de salida**, los niveles finales de **diésel**, **AdBlue** y **aceite**, lo que permite llevar un control administrativo del uso de las unidades y facilitar la detección de inconsistencias o anomalías en la operación.

Además, el sistema verifica si el operador tiene una jornada activa antes de permitir el cierre del registro, lo que ayuda a mantener la integridad de la información registrada en la base de datos.

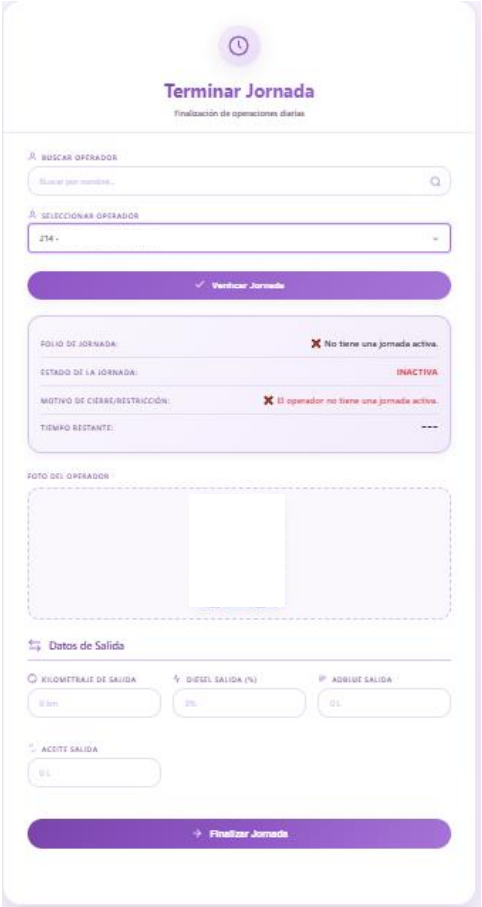

Figura 3.15 Finalización de jornada del operador.

3.3.3.3 Relevo de jornada

El sistema también contempla el proceso de **relevo de jornada**, el cual permite transferir la operación de una unidad de un operador saliente a un operador entrante. Este proceso es común en operaciones de transporte donde los turnos pueden dividirse entre diferentes operadores.

Durante el relevo se registran datos del operador que entrega la unidad y del operador que la recibe, así como información relacionada con el estado del vehículo al momento del cambio, incluyendo **kilometraje**, **niveles de combustible** y otros indicadores operativos.

Este mecanismo permite mantener continuidad en la operación del servicio y registrar adecuadamente la responsabilidad de cada operador sobre la unidad durante su turno.

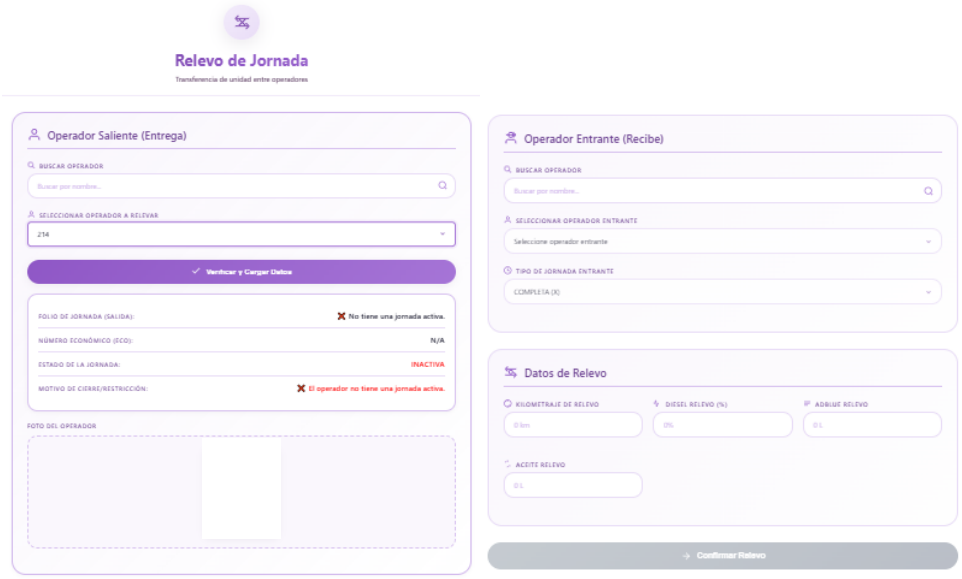

Figura 3.16 Proceso de relevo de jornada entre operadores.

3.3.4 Módulo de control de suministro de diésel

El sistema incluye un módulo destinado al **control del suministro de diésel**, el cual permite registrar y monitorear el consumo de combustible de las unidades del Corredor. Este módulo facilita el seguimiento de las cargas de combustible realizadas a los autobuses, permitiendo mantener un registro detallado de los litros suministrados y del estado del tanque durante la operación diaria.

Dentro de este módulo es posible registrar nuevas entradas de diésel asociadas a cada autobús, capturando información como el **número económico de la unidad (ECO)**, el **kilometraje del vehículo al momento de la carga** y la **cantidad de litros de diésel suministrados**. Esta información permite llevar un control más preciso del consumo de combustible y facilita la detección de posibles inconsistencias o variaciones en el rendimiento de las unidades.

Además, el sistema mantiene un **historial de registros de suministro**, donde se almacenan datos como la fecha, la hora, la unidad abastecida, el kilometraje y los litros suministrados. Este historial permite consultar de manera rápida las cargas de combustible realizadas durante el día.

Asimismo, el módulo incluye un apartado que muestra **indicadores acumulados de combustible**, tales como el total de litros suministrados a autobuses, el total de litros administrados a vehículos externos y el total general de combustible registrado en el sistema. Esta funcionalidad facilita el control administrativo del combustible dentro de la empresa.

Finalmente, el sistema permite realizar el **cierre del día**, registrando los litros finales disponibles en el tanque de almacenamiento. Este proceso permite llevar un control diario del inventario de combustible y facilita la conciliación de los registros de suministro realizados durante la jornada.

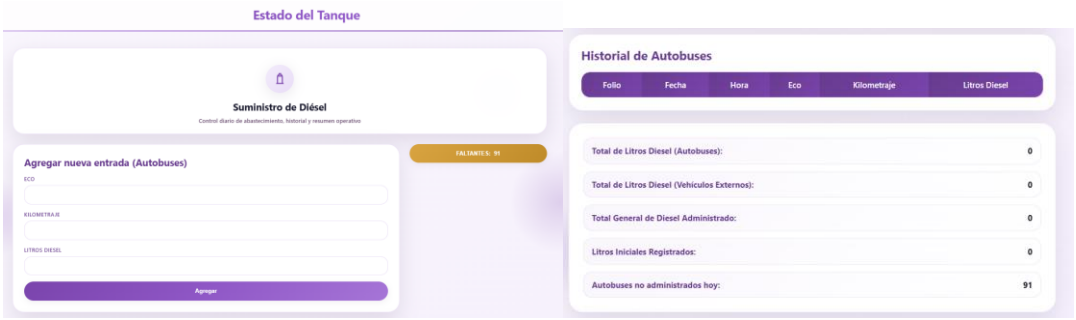

Figura 3.17 Módulo de control de suministro de diésel en el sistema web administrativo.

3.3.5 Módulo de monitoreo de jornadas activas (Relojes)

Dentro del apartado de telemetría, el sistema incorpora un módulo denominado “Relojes”, el cual permite visualizar en tiempo real el estado de las jornadas activas de los operadores que se encuentran en operación. Este módulo funciona como una herramienta de monitoreo que permite al personal de supervisión observar el estado actual de los operadores y de las unidades que se encuentran en servicio.

En esta interfaz se muestran todos los operadores que tienen una jornada activa, presentando información relevante como el nombre del operador, el número económico de la unidad (ECO), la hora de inicio de la jornada, el tipo de jornada asignada, la duración de la jornada y el tiempo restante para finalizar el turno. Esta información permite al personal de monitoreo tener una visión general del estado operativo del servicio en cada momento.

Adicionalmente, el sistema muestra indicadores relacionados con el estado de la operación, tales como la presencia de siniestros, el número de vueltas realizadas durante la jornada y otras acciones operativas que pueden aplicarse sobre cada operador.

Para el registro de ubicación en eventos relacionados con siniestros, se integró un mapa interactivo basado en OpenStreetMap mediante la biblioteca Leaflet. La elección de OpenStreetMap se realizó debido a que es una alternativa de software libre que permite utilizar información geográfica sin costos de licenciamiento, lo cual resultó adecuado para las condiciones del proyecto y permitió evitar gastos adicionales en servicios comerciales de mapas.

La integración del mapa se realiza desde el archivo HTML del módulo, donde se carga la hoja de estilos de Leaflet y posteriormente la biblioteca JavaScript necesaria para generar el mapa interactivo dentro de la interfaz.

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
<div id="map-siniestro" style="height: 200px;"></div>
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
```

El mapa se muestra dentro del formulario modal de registro de siniestro. Este formulario contiene un campo de búsqueda de ubicación, un campo de coordenadas, el contenedor del mapa y un botón para obtener la ubicación actual del usuario.

```
<input type="text" id="search-location-siniestro" placeholder="Ingresa una dirección">
<button type="button" onclick="searchLocationSiniestro()">Buscar</button> <input type="text" id="ubicacion"
readonly>
<div id="map-siniestro" style="height: 200px;"></div>
<button id="retry-geolocation-siniestro">Obtener mi ubicación</button>
```

La inicialización del mapa se realiza mediante la función **initMapSiniestro().** En esta función se define una ubicación inicial, se crea el mapa con **L.map(),** se agregan las capas visuales de OpenStreetMap mediante **L.tileLayer()** y se coloca un marcador interactivo sobre el mapa.

```
function initMapSiniestro() {
      const defaultLocation = [19.432608, -99.133209];
       mapSiniestro = L.map('map-siniestro').setView(defaultLocation, 12);
       L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
          attribution: '© OpenStreetMap contributors'
       }).addTo(mapSiniestro);
       markerSiniestro = L.marker(defaultLocation, {
          draggable: true
       }).addTo(mapSiniestro);
       txtUbicacion.value = `${defaultLocation[0]},${defaultLocation[1]}`;
}
```

La obtención de coordenadas se realiza de diferentes formas. La primera consiste en permitir que el usuario arrastre el marcador sobre el mapa. Cuando el marcador cambia de posición, el sistema obtiene la latitud y longitud mediante **getLatLng()** y coloca las coordenadas en el campo correspondiente.

```
markerSiniestro.on('dragend', () =>
   { const p = markerSiniestro.getLatLng();
   txtUbicacion.value = `${p.lat},${p.lng}`;
});
```

La segunda forma consiste en permitir que el usuario haga clic directamente sobre el mapa. Cuando ocurre este evento, el marcador se mueve al punto seleccionado y el sistema actualiza automáticamente el campo de coordenadas.

```
mapSiniestro.on('click', e => {
   markerSiniestro.setLatLng(e.latlng);
   txtUbicacion.value = `${e.latlng.lat},${e.latlng.lng}`;
});
```

Además, el sistema utiliza la API de geolocalización del navegador para intentar obtener la ubicación actual del usuario. Si el navegador permite el acceso a la ubicación, el mapa se centra en las coordenadas obtenidas y el marcador se posiciona automáticamente en ese punto.

```
if (navigator.geolocation) {
       navigator.geolocation.getCurrentPosition(
          pos => {
          const loc = [pos.coords.latitude, pos.coords.longitude]; mapSiniestro.setView(loc, 15);
```

```
            markerSiniestro.setLatLng(loc);
            txtUbicacion.value = `${loc[0]},${loc[1]}`;
        },
    () => alert('No se pudo obtener tu ubicación. Usa el mapa para seleccionar manualmente.'),
        {
            enableHighAccuracy: true,
            timeout: 10000,
            maximumAge: 0
        }
    );
}
```

También se incorporó una función de búsqueda de ubicación utilizando Nominatim, servicio asociado a OpenStreetMap que permite convertir una dirección escrita por el usuario en coordenadas geográficas. Para ello se construye una URL con el texto ingresado, se realiza una petición mediante **fetch()** y se procesa la respuesta en formato JSON.

```
window.searchLocationSiniestro = function() {
    const query = searchLocationInput.value.trim();
    if (!query) { return alert('Ingresa una dirección para buscar.'); }
    const url =
`https://nominatim.openstreetmap.org/search?format=json&q=${encodeURIComponent(query)}&limit=1`;
    fetch(url, {
        headers: {
            'User-Agent': 'SiniestrosApp/1.0'
        }
    }) .then(r => r.json())
    .then(data => {
        if (data && data.length > 0) {
            const loc = [
                parseFloat(data[0].lat),
                parseFloat(data[0].lon)
            ];
            mapSiniestro.setView(loc, 15);
            markerSiniestro.setLatLng(loc);
            txtUbicacion.value = `${loc[0]},${loc[1]}`;
        } else {
            alert('No se encontraron resultados para la dirección ingresada.');
        }
    });
};
```

Una vez seleccionada la ubicación, el sistema valida que el campo de coordenadas contenga un formato correcto de latitud y longitud antes de enviar la información al servidor. Si las coordenadas son válidas, se envían junto con el ECO, el operador, el estado del siniestro y la descripción del evento.

```
btnOkSiniestro.onclick = () => {
   const ubicacion = txtUbicacion.value.trim();
   const descripcion = txtDescSiniestro.value.trim();
   if (!ubicacion || !/^-?\d+(\.\d+)?,-?\d+(\.\d+)?$/.test(ubicacion))
      {
         return alert('Selecciona una ubicación válida en el mapa.');
   }
   enviarAccion({
      eco: ecoActual,
      id_chofer: idChoferActual,
      newStatus: 'EN',
      siniestro: {
         ubicacion, descripcion: descripcion || null
      }
   });
};
```

El envío de la información se realiza mediante una petición **POST** al archivo PHP correspondiente. Los datos se envían en formato JSON para que el backend pueda procesarlos, registrar el siniestro y almacenar la ubicación seleccionada.

```
function enviarAccion(payload, url = '../backend/reloj.php') {
   return fetch(url, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json'
   },
   body: JSON.stringify(payload)
})
   .then(r => r.json())
   .then(({ success, message }) => {
      if (!success) throw new Error(message);
      loadRows();
   });
}
```

Entre las funciones disponibles dentro del módulo se encuentran las siguientes:

• **Registro de siniestros:** permite indicar si una unidad ha estado involucrada en un incidente durante su operación. Cuando el estado cambia de “SIN” a “EN”, el sistema despliega un formulario para registrar la ubicación mediante el mapa interactivo, capturar las coordenadas geográficas y agregar una descripción del evento.

• **Control de tiempo de comida:** permite asignar al operador un periodo de descanso correspondiente al tiempo de comida, con una duración predeterminada de 25 minutos. Durante este periodo, el sistema registra temporalmente la pausa en la jornada y muestra un contador regresivo.

• **Desincorporación de unidad:** permite finalizar una jornada antes del tiempo programado cuando la operación determina que una unidad ya no es necesaria en servicio, aun cuando el operador tenga tiempo restante en su turno.

• **Registro de incidencias:** permite documentar situaciones en las que un operador incurre en alguna falta o incumple indicaciones del equipo de monitoreo. Estas incidencias quedan registradas para su seguimiento administrativo.

Gracias a estas funcionalidades, el módulo de relojes centraliza la supervisión de las jornadas activas, permite registrar eventos operativos con ubicación geográfica mediante OpenStreetMap y facilita la toma de decisiones en tiempo real dentro del sistema de transporte.

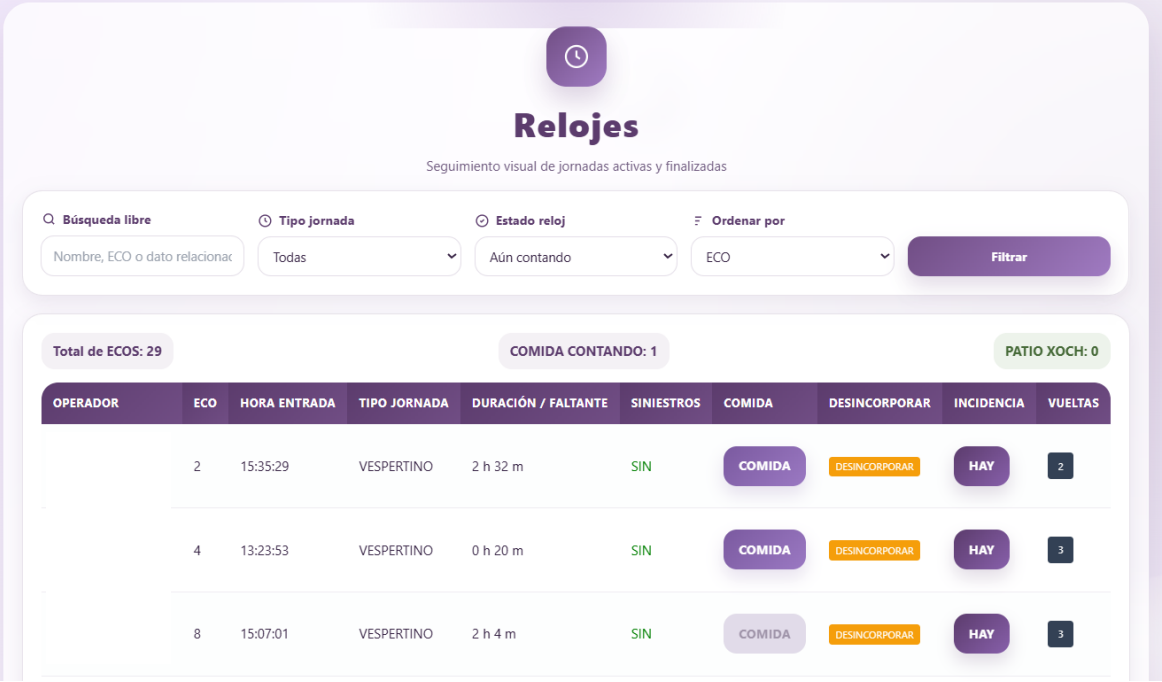

Figura 3.18 Módulo de monitoreo de jornadas activas (Relojes).

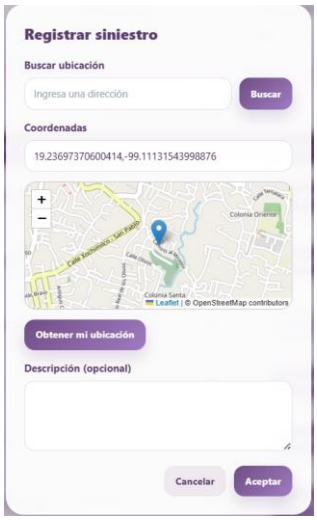

Figura 3.19 Registro de ubicación de siniestro mediante mapa interactivo basado en OpenStreetMap.

Figura 3.20 Registro de incidencia del operador

3.3.6 Módulo de gestión de siniestros

El sistema web administrativo incorpora un módulo destinado a la gestión y seguimiento de siniestros, el cual permite registrar incidentes ocurridos durante la operación de las unidades del Corredor. Este módulo facilita la documentación de los eventos, permitiendo registrar información relevante sobre el incidente, la ubicación donde ocurrió y el personal involucrado.

Cuando durante la operación se detecta un incidente desde el módulo de monitoreo de jornadas activas (Relojes), el personal de telemetría puede marcar la unidad como involucrada en un siniestro. En ese momento se despliega un formulario que permite seleccionar la ubicación del evento mediante un mapa interactivo basado en OpenStreetMap y registrar una descripción inicial de los hechos.

Las coordenadas geográficas obtenidas durante este primer registro son almacenadas en la base de datos PostgreSQL dentro de la tabla **siniestros**, específicamente en el campo **ubicacion**, junto con la información básica del evento.

```
Siniestros_activos
-----------------------------------------
folio
fecha
hora
eco
id_chofer
ubicacion
descripcion
estatus
...
```

De esta manera, cuando el área de siniestros accede posteriormente al expediente, ya no es necesario volver a capturar la ubicación, ya que ésta se recupera directamente desde la base de datos.

Cuando un usuario ingresa al módulo de siniestros pendientes, el sistema realiza una consulta al servidor para obtener todos los eventos registrados desde el módulo de telemetría que aún no han sido concluidos.

```
fetch("../backend/siniestros_formulario.php", {
   method: "POST",
   cache: "no-store"
})
.then((res) => res.text())
.then((text) => {
   const resp = JSON.parse(text);
   const rows = Array.isArray(resp) ? resp : resp.data;
});
```

La respuesta obtenida contiene información como el folio, número económico de la unidad, operador involucrado, descripción y ubicación geográfica. La ubicación es recuperada desde PostgreSQL como una cadena de texto compuesta por la latitud y longitud separadas por una coma.

```
const ubicacion = String(item.ubicacion ?? "").trim();
const [lat = "", lng = ""] = ubicacion.split(",");
```

A partir de estas coordenadas, el sistema genera dinámicamente un enlace hacia Google Maps para permitir al personal visualizar rápidamente el lugar exacto donde ocurrió el incidente.

```
function verUbicacion(lat, lng) {
   const url = `https://www.google.com/maps?q=${lat},${lng}`;
   window.open(url, "_blank");
}
```

Por ejemplo, si en la base de datos se encuentra almacenada la siguiente ubicación:

19.432608,-99.133209

el sistema construirá automáticamente la siguiente dirección:

https://www.google.com/maps?q=19.432608,-99.133209

Al seleccionar la opción **Ver ubicación**, Google Maps abre una nueva pestaña mostrando el punto exacto registrado previamente por el personal de telemetría.

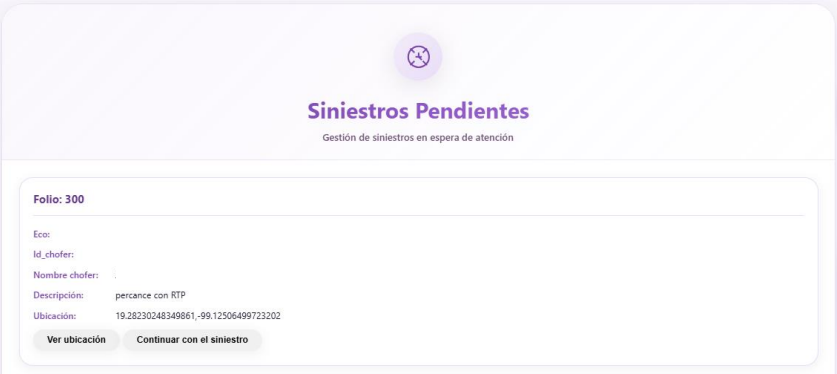

Figura 3.21 Vista de siniestros pendientes en el sistema web administrativo.

Además de visualizar la ubicación, el sistema permite continuar con el registro completo del siniestro mediante la opción **Continuar con el siniestro**. Para ello se construye una URL interna que envía la información básica del evento hacia el formulario **formulario_siniestro.html.**

```
function abrirVentanaSiniestro(
    folio,
    eco,
    id_chofer,
    nombre,
    descripcion,
    ubicacion
) {
    const url =
        `continuar_siniestro.html?folio=${folio}` +
        `&eco=${eco}` +
        `&id_chofer=${id_chofer}` +
        `&nombre=${nombre}` +
        `&descripcion=${descripcion}` +
        `&ubicacion=${ubicacion}`;

    window.open(url, "_blank", "width=900,height=700");
}
```

Una vez abierta la nueva ventana, JavaScript recupera los parámetros enviados mediante la URL y los asigna automáticamente a los controles del formulario.

```
function getQueryParam(name) {
    const url = new URL(window.location.href);
    const v = url.searchParams.get(name);
```

```
    return v ? decodeURIComponent(v) : "";
}
const folio = getQueryParam("folio");
const eco = getQueryParam("eco");
const id_chofer = getQueryParam("id_chofer");
const nombre = getQueryParam("nombre");
const descripcion = getQueryParam("descripcion");
const ubicacion = getQueryParam("ubicacion");
```

Posteriormente se inicializa un mapa interactivo basado en OpenStreetMap mediante la biblioteca Leaflet. La ubicación recuperada desde PostgreSQL se utiliza como punto inicial del mapa.

```
function initMap(ubicacion) {
    const defaultLocation = [19.432608, -99.133209];
    let initialLocation = defaultLocation;
    if (ubicacion) {
        const parts = ubicacion.split(",");
        if (parts.length === 2) {
            initialLocation = [
                parseFloat(parts[0]),
                parseFloat(parts[1])
            ];
        }
    }
    const map = L.map("map")
        .setView(initialLocation, 15);
    L.tileLayer(
        "https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png",
        {
            attribution:
            "© OpenStreetMap contributors"
        }
    ).addTo(map);
    const marker = L.marker(
        initialLocation,
        {
            draggable: true
        }
    ).addTo(map);
}
```

El usuario puede modificar la ubicación del incidente arrastrando el marcador o haciendo clic sobre otra zona del mapa. Cada movimiento actualiza automáticamente las coordenadas almacenadas en el formulario.

```
marker.on("dragend", function () {
    const position = marker.getLatLng();
    document.getElementById("ubicacion")
```

```
        .value =
        `${position.lat},${position.lng}`;
});
map.on("click", function (e) {
    const lat = e.latlng.lat;
    const lng = e.latlng.lng;
    marker.setLatLng([lat, lng]);
    document.getElementById("ubicacion")
        .value =
        `${lat},${lng}`;
});
```

El formulario permite complementar la información inicial agregando datos como:

- Personal responsable del seguimiento del siniestro.
- Determinación de responsabilidad del operador.
- Información relacionada con aseguradoras.
- Deducibles y gastos asociados.
- Fotografías de evidencia.
- Firma digital del operador.
- Descripción ampliada de los hechos.

Para la captura de la firma se utiliza un elemento HTML Canvas que permite dibujar directamente desde el navegador.

```
<canvas id="firmaCanvas"></canvas>
```

Una vez concluido el llenado del formulario, la firma se convierte en una imagen PNG y se adjunta al envío mediante **FormData.**

```
canvas.toBlob(function(blob) {
    const firmaFile =
        new File(
            [blob],
            "firma.png",
            {
                type: "image/png"
            }
        );
    formData.append(
        "firma_operador",
        firmaFile
    );
}, "image/png");
```

Finalmente, toda la información es enviada al archivo **formulario_siniestro.php** mediante una petición HTTP POST.

```
fetch(
  "../backend/formulario_siniestro.php",
  {
    method: "POST",
    body: formData
  }
);
```

Con esta implementación, el módulo de siniestros permite dar continuidad a los eventos registrados inicialmente desde telemetría, reutilizando las coordenadas almacenadas en PostgreSQL, integrando OpenStreetMap para la visualización y edición de la ubicación, generando enlaces dinámicos hacia Google Maps para consulta rápida y almacenando de manera centralizada toda la evidencia relacionada con cada incidente ocurrido durante la operación de las unidades.

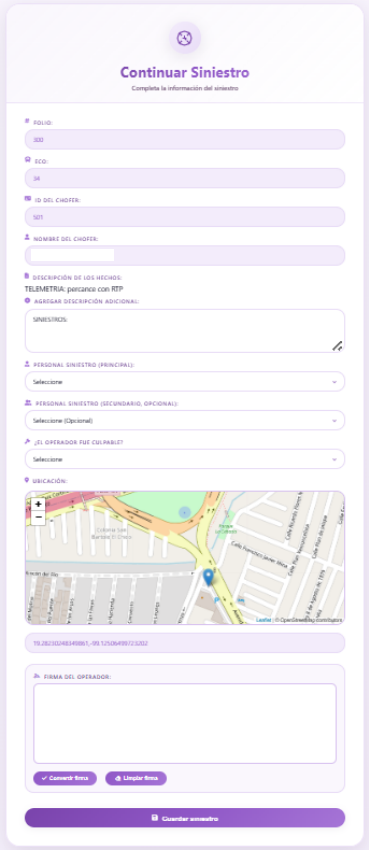

Figura 3.22 Registro detallado de siniestro en el sistema web administrativo.

3.3.7 Consultas y visualización de información

Además de los módulos de registro y control operativo, el sistema web administrativo incorpora diversas herramientas para la consulta, análisis y visualización de la información almacenada en la base de datos. Estas herramientas permiten al personal administrativo y operativo acceder rápidamente a los registros generados por el sistema y obtener información útil para la supervisión de la operación del Corredor.

Las interfaces de consulta fueron diseñadas mediante tablas dinámicas, filtros de búsqueda, indicadores visuales, mapas interactivos y gráficas, lo que facilita localizar información específica sobre operadores, unidades, jornadas laborales, consumo de combustible, vigencia documental y eventos registrados durante la operación diaria.

Dentro de estas herramientas se incluyen módulos de analítica, visualización geográfica y reportes administrativos, los cuales permiten interpretar los datos generados por el sistema de manera clara y organizada.

Analítica de siniestros

El sistema incluye un módulo de visualización geográfica que permite identificar las zonas con mayor registro de siniestros dentro del recorrido del Corredor. Esta funcionalidad utiliza mapas basados en OpenStreetMap mediante la biblioteca Leaflet, donde se representan gráficamente los puntos geográficos de los incidentes registrados.

La interfaz HTML carga la biblioteca Leaflet y define un contenedor donde se renderiza el mapa.

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css" />
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>
<div id="map"></div>
```

El mapa se inicializa desde JavaScript con una ubicación predeterminada correspondiente a la Ciudad de México. Posteriormente se agrega la capa visual de OpenStreetMap.

```
function initMap() {
  const defaultLocation = [19.432608, -99.133209];
  map = L.map('map').setView(defaultLocation, 12);
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '© OpenStreetMap contributors'
  }).addTo(map);
}
```

Para cargar los puntos de siniestros, el sistema realiza una petición al backend mediante **fetch().** El archivo PHP consulta los registros almacenados en PostgreSQL y devuelve las coordenadas junto con el número de siniestros registrados en cada zona.

```
function loadAccidentZones() {
  fetch('../backend/siniestro_zona.php', { method: 'POST' })
    .then(response => response.text())
    .then(text => {
      const data = JSON.parse(text);

      if (!data.success) {
        alert(`Error al cargar zonas de siniestros: ${data.message}`);
        return;
      }

      data.zones.forEach(zone => {
        const lat = parseFloat(zone.lat);
        const lng = parseFloat(zone.lng);
        const count = parseInt(zone.accident_count, 10);
      });
    });
}
```

Cada punto se coloca sobre el mapa mediante **L.circleMarker().** En este caso se utiliza un marcador circular de color rojo para representar visualmente las zonas donde existen registros de siniestros. Cada punto incluye una ventana emergente que muestra el número de incidentes asociados a esa ubicación.

```
L.circleMarker([lat, lng], {
  radius: 10,
  color: 'red',
  fillColor: 'red',
  fillOpacity: 0.5,
  weight: 1
})
.addTo(map)
.bindPopup(`Siniestros en esta zona: <b>${count}</b>`);
```

Los puntos se identifican mediante las coordenadas geográficas obtenidas desde la base de datos. Al recorrer los registros, el sistema convierte los valores de latitud y longitud a datos numéricos, valida que sean coordenadas válidas y posteriormente los agrega al arreglo puntos.

```
const puntos = [];
data.zones.forEach(zone => {
  const lat = parseFloat(zone.lat);
  const lng = parseFloat(zone.lng);
  if (isNaN(lat) || isNaN(lng)) return;
  puntos.push([lat, lng]);
});
```

Una vez colocados los marcadores, el sistema ajusta automáticamente la vista del mapa para mostrar todos los puntos registrados. Esto se realiza mediante **L.latLngBounds()** y **map.fitBounds().**

```
if (puntos.length > 0) {
  const bounds = L.latLngBounds(puntos);
  if (bounds.isValid()) {
    map.fitBounds(bounds, { padding: [50, 50] });
  }
}
```

Gracias a esta visualización, el personal puede identificar patrones de ocurrencia y zonas donde se concentra un mayor número de incidentes, lo que resulta útil para el análisis operativo y la toma de decisiones relacionadas con la seguridad del servicio.

Figura 3.23 Visualización geográfica de zonas con mayor registro de siniestros.

Consulta de información de jornadas laborales

El sistema permite consultar el historial de jornadas laborales de los operadores mediante una interfaz que muestra información detallada sobre cada registro. En esta vista se presentan datos como el número de empleado, nombre del operador, unidad asignada, fecha de entrada, hora de entrada, fecha de salida, hora de salida, tipo de jornada, duración, horas faltantes y vueltas realizadas.

La búsqueda se realiza mediante filtros de operador y rango de fechas. Cuando el usuario ejecuta la consulta, JavaScript construye una URL con los parámetros seleccionados y realiza una petición al archivo PHP correspondiente.

```
function searchJornada() {
  const id = document.getElementById('id_chofer').value;
  const fi = document.getElementById('start_date').value;
  const ff = document.getElementById('end_date').value;
```

```
fetchJSON(`../backend/obtencion_de_jonada.php?id_chofer=${encodeURIComponent(id)}&start_date=${encod
eURIComponent(fi)}&end_date=${encodeURIComponent(ff)}`)
    .then(d => {
      currentResults = d.data;
      displayResults(fi, ff, id);
    });
}
```

La información obtenida se almacena temporalmente en el arreglo **currentResults**, el cual es utilizado para construir la tabla dinámica en la interfaz.

```
currentResults.forEach(r => {
  const esperado = journeyHours[(r.jornada_h || '').trim()] || 0;
  const trabajado = Number(r.duration_seconds || 0);
});
```

El sistema calcula la duración de la jornada y la compara contra las horas esperadas de acuerdo con el tipo de jornada. Para ello se utiliza una estructura que define las horas correspondientes a cada tipo.

```
const journeyHours = {
  'MATUTINO': 10 * 3600,
  'VESPERTINO': 10 * 3600,
  'MIXTO': 10 * 3600,
  'COMPLETA': 16 * 3600,
  '8': 8 * 3600,
  '10': 10 * 3600,
  '11': 11 * 3600,
  '12': 12 * 3600
};
```

Posteriormente, se asigna una clase visual dependiendo del cumplimiento de la jornada. Esto permite identificar mediante colores si la jornada fue cumplida, si tiene una diferencia menor o si presenta un faltante significativo.

```
function durationClass(sec, tipo) {
  const esperado = journeyHours[(tipo || '').trim()];
  if (sec >= esperado) return 'duration-match';

  const diff = esperado - sec;
  if (diff > 60 && diff <= 3600) return 'duration-warning';
  if (diff > 3600) return 'duration-exceed';
  return '';
}
```

La tabla se construye dinámicamente mediante JavaScript, agregando las clases correspondientes a cada celda para representar visualmente el estado de la jornada.

```
html += `<tr>
  <td>${r.eco ?? ''}</td>
  <td>${r.fecha_entrada ?? ''}</td>
  <td>${r.hora_entrada ?? ''}</td>
  <td>${r.fecha_salida ?? ''}</td>
  <td>${r.hora_salida ?? ''}</td>
  <td>${r.jornada_h ?? ''}</td>
  <td class="${durClass}">${durText}</td>
  <td>${toHHMMSS(faltante)}</td>
</tr>`;
```

Además de la consulta individual, el sistema genera listas semanales agrupando los registros por operador y por día. Para esto se construye un objeto donde cada operador contiene sus jornadas organizadas por fecha.

```
const map = {};
res.forEach(r => {
  const nom = `${r.id_chofer} - ${r.nombre} ${r.apellido_paterno} ${r.apellido_materno}`;
  if (!map[nom]) {
    map[nom] = { id: r.id_chofer, journeys: {} };
  }
  if (!map[nom].journeys[r.fecha_entrada]) {
    map[nom].journeys[r.fecha_entrada] = [];
  }
  map[nom].journeys[r.fecha_entrada].push(r);
});
```

Esta información permite construir reportes semanales, identificar faltas, descansos, vacaciones, incapacidades y jornadas desincorporadas. Cada caso se representa mediante una clase visual distinta.

```
if (hayVacaciones) {
  cellText = 'VACACIONES';
  cellClass = 'duration-vacaciones';
} else if (hayDescanso) {
  cellText = 'DESCANSO';
  cellClass = 'duration-descanso';
} else if (hayFalta) {
  cellText = 'FALTA';
  cellClass = 'duration-falta';
}
```

El sistema también permite exportar los reportes a PDF utilizando **jsPDF** y **autoTable**. Durante la generación del PDF, las clases visuales son interpretadas y convertidas a colores dentro del documento.

```
const colorMap = {
  'duration-match': [155, 245, 155],
  'duration-warning': [255, 217, 102],
  'duration-exceed': [255, 118, 118],
  'duration-desincorporado': [130, 202, 255],
  'duration-falta': [242, 104, 19],
  'duration-descanso': [103, 41, 158],
  'duration-vacaciones': [237, 145, 237]
};
if (color) {
  data.cell.styles.fillColor = color;
  data.cell.styles.textColor = textColor;
}
```

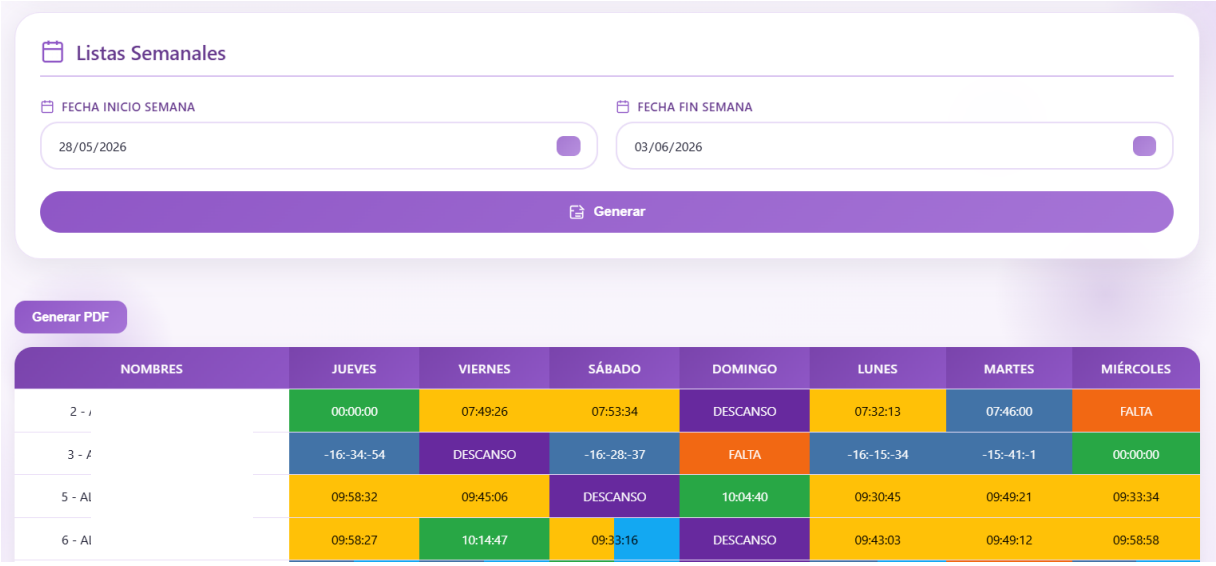

Figura 3.24 Consulta de información de jornadas laborales

Consulta de estatus de seguros de unidades

Dentro del sistema también se implementó un módulo para el seguimiento de la vigencia de los seguros de los autobuses. Este módulo permite visualizar el estado actual del seguro de cada unidad, indicando si la póliza se encuentra vigente, próxima a vencer o vencida.

La información se obtiene desde el backend mediante una petición **POST** al archivo **aseguradoras_autobuses.php.**

```
const formData = new FormData();
formData.append("action", "load");
const response = await fetch("../backend/aseguradoras_autobuses.php", {
  method: "POST",
```

```
  body: formData,
  credentials: "same-origin"
});
```

Una vez recibida la información, el sistema compara la fecha de vigencia del seguro con la fecha actual. Si la fecha ya venció, se asigna la clase red; si vence dentro del siguiente mes, se asigna la clase yellow; y si aún tiene más de treinta días de vigencia, se asigna la clase green.

```
let hoy = new Date();
hoy.setHours(0, 0, 0, 0);
let unMesDespues = new Date(hoy);
unMesDespues.setMonth(unMesDespues.getMonth() + 1);
if (fechaSeguro < hoy) {
  clase = "red";
} else if (fechaSeguro <= unMesDespues) {
  clase = "yellow";
} else {
  clase = "green";
}
```

Cada fila de la tabla recibe la clase correspondiente, permitiendo identificar visualmente el estado del seguro.

```
const tr = document.createElement("tr");
tr.className = clase;
tdEco.textContent = auto.eco ?? "";
tdPlaca.textContent = auto.placa ?? "";
tdAseguradora.textContent = auto.nombre_aseguradora ?? "";
tdSeguro.textContent = auto.fecha_vigencia_seguro ?? "";
tdEstatus.textContent = auto.estatus_fecha_seguro ?? "";
```

La interfaz incluye una leyenda visual para explicar el significado de cada color: verde para seguro vigente, amarillo para seguro próximo a vencer y rojo para seguro vencido.

Figura 3.25 Consulta del estatus de seguro de autobuses

Analítica de consumo de diésel

El sistema también incorpora herramientas de análisis relacionadas con el consumo de combustible de las unidades. Mediante el módulo de analítica de diésel, es posible consultar los litros suministrados a una unidad específica dentro de un rango de fechas determinado.

La interfaz carga la biblioteca Chart.js, la cual se utiliza para generar gráficas dinámicas a partir de la información obtenida desde PostgreSQL.

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<canvas id="dieselChart"></canvas>
```

El usuario selecciona el número económico de la unidad y el rango de fechas. Posteriormente JavaScript construye los parámetros de consulta y realiza una petición al backend.

```
const params = new URLSearchParams({
  eco,
  fechaInicio,
  fechaFin
});
fetch("../backend/get_analitica_diesel.php?" + params.toString())
  .then(r => r.json())
  .then(data => {
    mostrarGrafica(data.datos);
  });
```

La información recibida se presenta en una tabla, donde se muestra la fecha y los litros suministrados. Al mismo tiempo, el sistema calcula el total de litros suministrados y los días en los que no se registró abastecimiento.

```
let totalLitros = 0;
let zeroLitrosCount = 0;
data.datos.forEach(d => {
  const litros = parseFloat(d.litros) || 0;
  totalLitros += litros;
```

```
  if (litros === 0) {
    zeroLitrosCount++;
  }
});
```

Estos indicadores se muestran al usuario como resumen del periodo consultado.

```
summaryText.innerHTML = `
  <strong>Total Litros Suministrados: ${totalLitros.toFixed(2)}</strong><br>
  <strong>Días no Suministrados: ${zeroLitrosCount}</strong>
`;
```

Para generar la gráfica, el sistema separa las fechas y los litros en dos arreglos. Las fechas se utilizan como etiquetas del eje X y los litros como valores del eje Y.

```
const labels = datos.map(d => d.fecha);
const values = datos.map(d => parseFloat(d.litros) || 0);
```

Finalmente, se crea una gráfica de tipo línea mediante **Chart.js**.

```
dieselChart = new Chart(ctx, {
  type: "line",
  data: {
    labels,
    datasets: [{
      label: "Litros Suministrados",
      data: values,
      borderColor: "blue",
      backgroundColor: "rgba(0, 0, 255, 0.1)",
      fill: true
    }]
  },
  options: {
    scales: {
      x: {
        title: {
          display: true,
          text: "Fecha"
        }
      },
      y: {
        beginAtZero: true,
        title: {
          display: true,
          text: "Litros"
        }
      }
```

```
    }
  }
});
```

Antes de generar una nueva gráfica, el sistema destruye la anterior para evitar que se dupliquen los datos en pantalla.

```
if (dieselChart) {
  dieselChart.destroy();
}
```

Esta visualización permite observar el comportamiento del consumo de combustible a lo largo del tiempo, identificar días sin suministro y analizar el consumo por unidad.

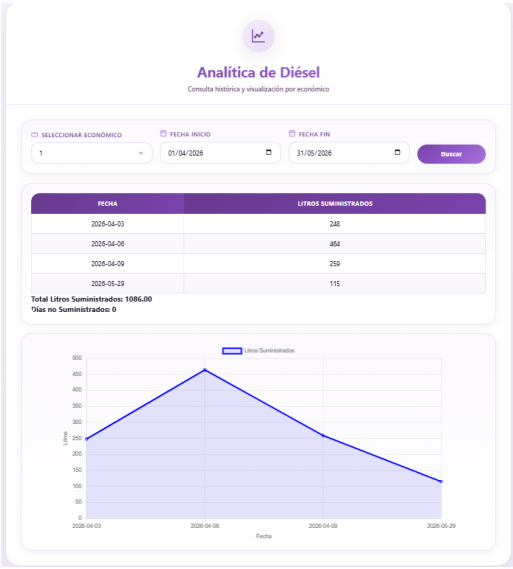

Figura 3.26 Módulo de analítica de consumo de diésel.

Consulta de vigencia de tarjetones de operadores

El sistema también permite consultar la vigencia de los tarjetones de los operadores, documento necesario para la operación del servicio de transporte. Este módulo muestra la fecha de vencimiento del tarjetón de cada operador, permitiendo identificar aquellos que requieren renovación próxima.

La información se obtiene mediante una petición al archivo PHP correspondiente, el cual devuelve los datos de los operadores junto con un valor de color calculado según el estado de vencimiento del tarjetón.

```
const formData = new FormData();
formData.append("action", "load");
const response = await fetch("../backend/listadodeoperdaores.php", {
  method: "POST",
  body: formData,
  credentials: "same-origin"
```

```
});
```

Una vez recibidos los datos, JavaScript construye dinámicamente las filas de la tabla. El campo de vencimiento del tarjetón recibe una clase de color enviada por el backend.

```
(data.empleados || []).forEach((emp) => {
  const tr = document.createElement("tr");
  tr.innerHTML = `
    <td>${emp.id_chofer ?? ""}</td>
    <td>${emp.nombre ?? ""}</td>
    <td>${emp.apellido_materno ?? ""}</td>
    <td>${emp.apellido_paterno ?? ""}</td>
    <td class="${emp.color ?? ""}">
      ${emp.fecha_vencimiento_tarjeton ?? ""}
    </td>
  `;
  tbody.appendChild(tr);
});
```

La interfaz incluye una leyenda visual que permite interpretar los colores utilizados: verde para documentos vigentes, amarillo para documentos próximos a vencer y rojo para documentos vencidos.

De esta manera, el sistema no solamente almacena información operativa, sino que también transforma los datos registrados en herramientas visuales de análisis. Las tablas, mapas, colores e indicadores permiten consultar información de forma rápida, identificar situaciones críticas y apoyar la toma de decisiones administrativas y operativas dentro del Corredor.

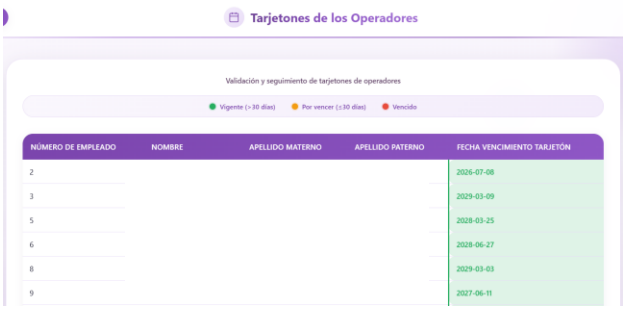

Figura 3.27 Consulta de vigencia de tarjetones de operadores.

3.4 Back-End

El *back-end* del sistema se encargó de implementar la lógica de negocio y gestionar la interacción entre la interfaz de usuario y la base de datos. Para este propósito se utilizó PHP como lenguaje de programación del lado del servidor, el cual permitió procesar las solicitudes realizadas por los usuarios, ejecutar las reglas de negocio de la aplicación y realizar operaciones sobre la base de datos PostgreSQL. La aplicación fue desplegada en un entorno Windows utilizando Internet Information Services (IIS) como servidor web.

Las funciones principales del back-end incluyen el procesamiento de formularios, la validación de datos, el manejo de solicitudes HTTP y la ejecución de consultas a la base de datos. Cada módulo

administrativo cuenta con su propia lógica de procesamiento, lo que permitió mantener una estructura de código organizada y modular.

Comunicación entre Front-End y Back-End

La comunicación entre la interfaz web y el servidor se implementó mediante solicitudes HTTP utilizando principalmente el método POST. Este método fue seleccionado debido a que permite enviar información capturada por el usuario sin exponerla directamente en la URL del navegador, además de facilitar el envío de estructuras complejas de datos y archivos adjuntos.

El flujo general de funcionamiento del sistema es el siguiente:

1. El usuario captura información en un formulario HTML5.
2. JavaScript realiza validaciones preliminares del lado del cliente.
3. Los datos son enviados mediante una petición HTTP POST.
4. PHP recibe la solicitud y valida la información recibida.
5. Se ejecutan consultas SQL en PostgreSQL.
6. Los resultados son convertidos a formato JSON.
7. JavaScript procesa la respuesta y actualiza dinámicamente la interfaz.

Un ejemplo simplificado de una petición realizada desde el navegador es el siguiente:

```
fetch("../backend/aseguradoras_autobuses.php", {
    method: "POST",
    body: formData,
    credentials: "same-origin"
})
.then(response => response.json())
.then(data => {
    console.log(data);
});
```

Posteriormente, PHP recibe la solicitud y ejecuta las consultas correspondientes sobre PostgreSQL:

```
$stmt = $pdo->prepare($sql);
$stmt->execute();
$resultados = $stmt->fetchAll(PDO::FETCH_ASSOC);
echo json_encode($resultados);
```

Este mecanismo fue utilizado en todos los módulos desarrollados dentro del sistema, incluyendo operadores, unidades, jornadas laborales, recaudo, diésel, mantenimiento, siniestros y telemetría.

Diseño y normalización de la base de datos

La base de datos fue desarrollada utilizando PostgreSQL debido a su estabilidad, rendimiento, capacidad para manejar relaciones complejas y soporte para integridad referencial mediante llaves primarias y foráneas.

Durante el diseño de la base de datos se aplicaron técnicas de normalización con el objetivo de evitar redundancia, inconsistencias y anomalías durante las operaciones de inserción, actualización y consulta de información.

El modelo relacional fue diseñado hasta la Tercera Forma Normal (3FN), garantizando una estructura organizada y escalable.

Primera Forma Normal (1FN)

Se asegura que cada atributo almacene únicamente valores atómicos y que no existan grupos repetitivos dentro de una misma tabla.

Por ejemplo, en la tabla empleado_chofer cada campo almacena un único dato específico, como nombre, CURP, RFC o fecha de vencimiento del tarjetón.

Segunda Forma Normal (2FN)

Todos los atributos dependen completamente de la clave primaria de su tabla.

Por ejemplo, la información personal de los operadores se almacena únicamente en la tabla empleado_chofer, mientras que las jornadas laborales almacenan únicamente el identificador del operador mediante el campo id_chofer.

De esta forma se evita duplicar información en múltiples tablas.

Tercera Forma Normal (3FN)

No existen dependencias transitivas entre atributos no clave.

La información de operadores, unidades, jornadas laborales, recaudo, mantenimiento, diésel y siniestros se encuentra distribuida en tablas independientes que se relacionan mediante llaves foráneas.

Gracias a esta estructura, una modificación realizada en la información de un operador o de una unidad se actualiza automáticamente en todas las consultas relacionadas sin necesidad de duplicar registros.

La aplicación de la Tercera Forma Normal permitió:

• Reducir la redundancia de información.

• Evitar inconsistencias durante actualizaciones.

• Mejorar la integridad referencial.

• Facilitar la generación de reportes.

• Optimizar el mantenimiento del sistema.

• Permitir la escalabilidad futura de la base de datos.

Ejemplo de consulta relacional utilizada en el sistema

Una de las consultas utilizadas dentro del sistema consiste en obtener la información completa de las jornadas laborales, relacionando operadores y unidades de transporte.

Para ello se emplean consultas SQL que integran información proveniente de múltiples tablas relacionadas mediante llaves foráneas:

```
SELECT
   h.folio,
   h.fecha_entrada,
   h.hora_entrada,
   h.fecha_salida,
   h.hora_salida,
   h.jornada_h,
   e.id_chofer,
   e.nombre,
   e.apellido_paterno,
   e.apellido_materno,
   a.eco,
   a.placa,
   a.tipo,
   h.kilometraje_entrada,
   h.kilometraje_salida,
   h.diesel_entrada,
   h.diesel_salida
FROM obtencion_de_jornadas h
INNER JOIN operadores e
   ON h.id_chofer = e.id_chofer
INNER JOIN autobuses a
   ON h.eco = a.eco
ORDER BY h.fecha_entrada DESC;
```

Esta consulta permite recuperar en una sola operación información distribuida en tres entidades principales del sistema:

• Operadores (empleado_chofer).

• Unidades de transporte (autobus).

• Jornadas laborales (historial_jornada_completa).

La necesidad de utilizar relaciones entre tablas demuestra la correcta aplicación de la Tercera Forma Normal, ya que los datos no se encuentran duplicados dentro de una sola tabla, sino organizados en entidades independientes relacionadas mediante llaves primarias y foráneas.

Gracias a este diseño, el sistema puede generar reportes completos de jornadas laborales, asociando automáticamente al operador correspondiente con la unidad utilizada durante la operación, manteniendo la consistencia de la información y reduciendo significativamente la posibilidad de errores derivados de registros duplicados.

En la **Figura 3.27** se presenta el modelo relacional de la base de datos diseñada para el sistema web administrativo. Dicho modelo muestra la estructura de las tablas, sus atributos principales y las relaciones existentes entre ellas, permitiendo organizar y mantener la información de manera consistente y centralizada.

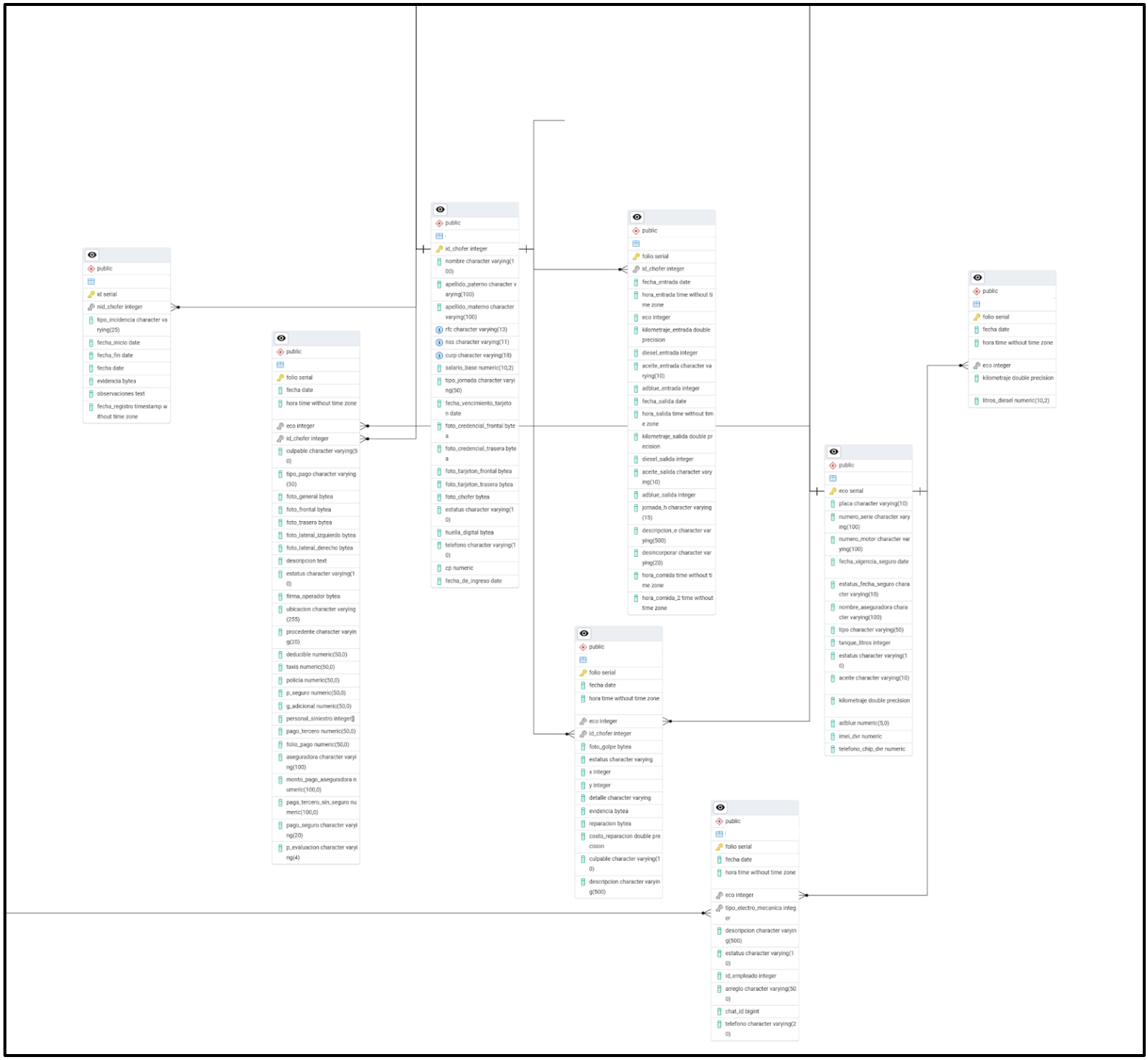

Figura 3.28 Modelo relacional de la base de datos del sistema administrativo.

3.5 Integración del sistema

La integración del sistema web administrativo se realizó mediante la interacción coordinada entre la interfaz de usuario (*Front-End*), la lógica del servidor (*Back-End*) y la base de datos PostgreSQL, permitiendo el procesamiento y almacenamiento centralizado de la información generada durante la operación del servicio de transporte.

El sistema fue desarrollado bajo una arquitectura cliente-servidor, en la cual el usuario interactúa con la aplicación a través de una interfaz web accesible desde un navegador. Las solicitudes generadas por el usuario son enviadas al servidor mediante peticiones HTTP, las cuales son procesadas por scripts desarrollados en PHP. Estos scripts se encargan de validar los datos recibidos, ejecutar la lógica de negocio correspondiente y realizar las operaciones necesarias sobre la base de datos.

La base de datos PostgreSQL actúa como el repositorio central de información del sistema, almacenando los registros relacionados con operadores, unidades, jornadas laborales, cargas de combustible, incidencias, siniestros y otros elementos administrativos. A través de consultas SQL ejecutadas desde el *back-end*, el sistema permite registrar nueva información, actualizar registros existentes y recuperar datos para su visualización en la interfaz.

La comunicación entre el *Front-End* y el *Back-End* se realiza mediante solicitudes asíncronas utilizando la Fetch API, lo que permite enviar y recibir información sin necesidad de recargar la página. Este mecanismo mejora la experiencia de uso del sistema y permite una interacción más dinámica con la plataforma.

Dentro del proceso de integración también se incorporaron herramientas externas que amplían las funcionalidades del sistema. Entre ellas destaca la utilización de OpenStreetMap y la librería Leaflet, las cuales permiten visualizar mapas interactivos para la localización de eventos como siniestros o incidencias registradas durante la operación de las unidades.

Asimismo, el sistema integra módulos de monitoreo operativo que permiten supervisar en tiempo real las jornadas laborales activas de los operadores, así como registrar eventos relacionados con la operación diaria, tales como descansos, incidencias o siniestros. Esta integración permite centralizar la información operativa y facilitar la toma de decisiones por parte del personal administrativo y de monitoreo.

El flujo general de funcionamiento del sistema puede resumirse en las siguientes etapas:

1. El usuario ingresa información mediante la interfaz web.
2. El navegador envía la solicitud al servidor mediante una petición HTTP.
3. El servidor procesa la solicitud utilizando scripts PHP.
4. El servidor ejecuta consultas o actualizaciones en la base de datos PostgreSQL.
5. El servidor devuelve una respuesta al *Front-End* en formato JSON.
6. La interfaz web procesa la respuesta y actualiza la información mostrada al usuario.

Este modelo de integración permite mantener una arquitectura modular, escalable y fácil de mantener, ya que cada componente del sistema cumple una función específica dentro del proceso de gestión de

la información. Además, facilita la incorporación de nuevas funcionalidades en futuras versiones del sistema.

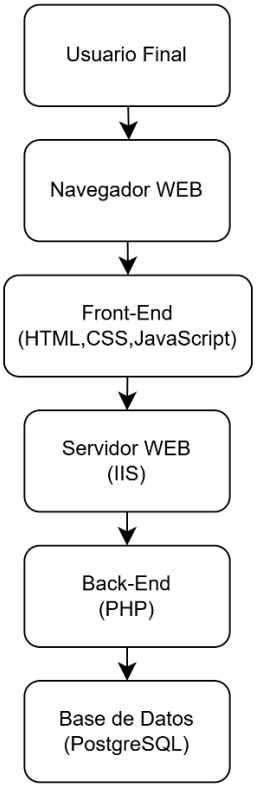

Figura 3.29 Arquitectura de integración del sistema web administrativo.
Fuente: Elaboración propia.

Capítulo IV. Resultados

Introducción

El presente capítulo describe los **resultados obtenidos tras la implementación del sistema web de gestión administrativa en el Corredor**, enfocándose en los beneficios operativos y administrativos derivados del uso de la solución tecnológica. A partir de la puesta en operación del sistema, se evaluó su impacto en la eficiencia de los procesos, el control de la información y el apoyo a la toma de decisiones dentro de la empresa.

4.1 Beneficios operativos y administrativos

La implementación del sistema web permitió centralizar la información administrativa en una sola plataforma digital, sustituyendo el manejo de registros manuales, cuadernos físicos y hojas de cálculo dispersas utilizadas anteriormente por las distintas áreas del Corredor.

Antes de la implementación del sistema, gran parte de la información debía buscarse físicamente en carpetas, hojas de registro almacenadas en autobuses, cuadernos de control o archivos de Excel distribuidos entre diferentes equipos de cómputo. Este proceso generaba retrasos considerables para localizar información, además de incrementar el riesgo de pérdida, duplicidad o inconsistencias en los registros.

Con la implementación del sistema web, toda la información quedó centralizada en una única base de datos PostgreSQL, permitiendo consultas inmediatas desde cualquier módulo autorizado de la plataforma.

Matriz comparativa de tiempos de operación

La Tabla 4.1 muestra una estimación comparativa de los tiempos promedio requeridos para realizar distintas actividades administrativas antes y después de la implementación del sistema.

Actividad	Antes del sistema	Después del sistema	Reducción aproximada
Búsqueda de expediente de operador	15 – 30 min	10 – 20 seg	98%
Consulta de historial de jornadas	20 – 40 min	5 – 15 seg	99%
Consulta de siniestros anteriores	30 – 60 min	10 – 30 seg	99%
Verificación de vigencia de tarjetones	10 – 20 min	5 seg	99%
Consulta de mantenimiento de una unidad	15 – 30 min	10 seg	98%

Consulta de cargas de diésel	10 – 20 min	5 seg	99%
Generación de reportes administrativos	1 – 2 horas	1 – 3 min	97%
Identificación de operadores activos	10 – 15 min	Tiempo real	100%

Tabla 4.1 comparativa de tiempos operativos antes y después de la implementación del sistema

Además de la reducción en tiempos de consulta, el sistema permitió disminuir significativamente la dependencia de documentos físicos, eliminando gran parte de los problemas asociados con extravío de hojas, deterioro por humedad, errores de captura manual y duplicidad de registros.

Comparativa porcentual de mejora

Con base en los tiempos promedio observados durante la operación diaria, se estimó la mejora obtenida mediante la automatización de procesos administrativos.

Proceso	Mejora estimada
Localización de información	98%
Consulta de historiales	99%
Control documental	95%
Seguimiento de siniestros	99%
Control de combustible	98%
Gestión de jornadas laborales	97%
Generación de reportes	97%
Disponibilidad de información	100%

Tabla 4.2 Porcentaje estimado de mejora operativa obtenida mediante el sistema web administrativo

Automatización del cálculo de jornadas laborales

Uno de los beneficios más importantes obtenidos fue la automatización del cálculo de jornadas laborales. Antes de la implementación del sistema, el cálculo de horas trabajadas se realizaba manualmente utilizando hojas físicas almacenadas dentro de los autobuses o en archivos de control administrativo. Este procedimiento requería revisar registros escritos a mano, interpretar horarios y realizar cálculos manuales para determinar las horas efectivamente trabajadas por cada operador.

Con el sistema desarrollado, la duración de la jornada se calcula automáticamente a partir de la fecha y hora de entrada registradas al inicio de la operación y la fecha y hora de salida capturadas al finalizar la jornada.

La duración total se obtiene mediante la diferencia entre ambos registros:

Duración de Jornada = Hora de Salida − Hora de Entrada

Por ejemplo:

Hora Entrada	Hora Salida	Jornada Calculada
05:00	15:00	10 horas
06:00	22:00	16 horas
07:00	18:00	11 horas

Tabla 4.3 de calculo de tiempo de jornadas

Posteriormente, el sistema compara automáticamente las horas trabajadas contra el tipo de jornada asignado al operador (8, 10, 11, 12 o 16 horas), calculando las horas faltantes o excedentes.

Esta funcionalidad permitió:

- Eliminar errores de cálculo manual.
- Reducir tiempos de revisión de nómina.
- Obtener información inmediata sobre el cumplimiento de jornadas.
- Identificar operadores con jornadas incompletas.
- Detectar incidencias operativas en tiempo real.
- Facilitar la generación de reportes para Recursos Humanos y Operaciones.

Por ejemplo, cuando un operador tiene asignada una jornada de 10 horas y únicamente registra 8 horas efectivas de trabajo, el sistema calcula automáticamente un faltante de 2 horas y lo muestra mediante indicadores visuales dentro del historial de jornadas. De manera similar, si el operador supera el tiempo establecido, el sistema registra el excedente correspondiente.

Asimismo, el sistema incorpora diferenciación visual mediante colores para identificar rápidamente jornadas completas, jornadas con faltantes, vacaciones, descansos, incapacidades y otras situaciones especiales. Esto facilita la interpretación de la información y permite a las áreas administrativas tomar decisiones de manera más rápida y precisa.

En conjunto, estos elementos evidencian que el sistema no solamente digitalizó los procesos existentes, sino que optimizó significativamente la operación administrativa al proporcionar información centralizada, accesible y confiable en tiempo real, reduciendo tiempos de consulta y mejorando la calidad de la información utilizada para la toma de decisiones.

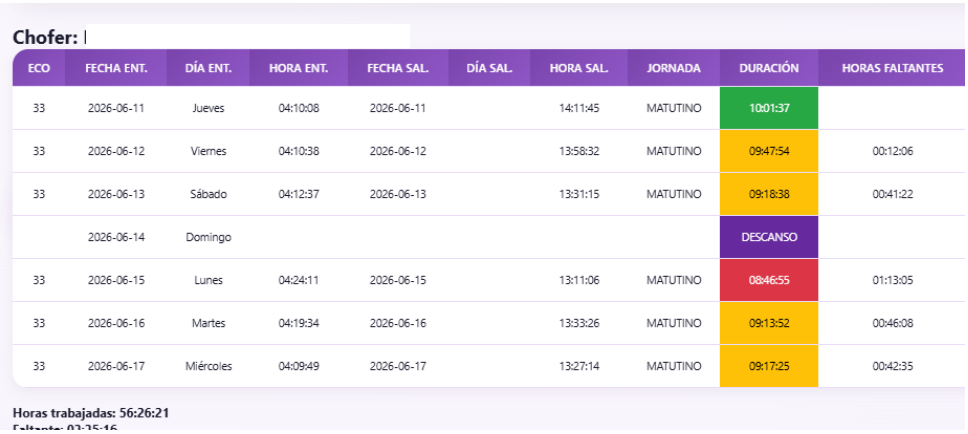

Figura 4.1 Historial Jornada del Operador

4.2 Impacto en la toma de decisiones

El acceso centralizado y oportuno a la información facilitó la toma de decisiones administrativas, al permitir consultar de forma rápida datos actualizados sobre operadores, unidades y eventos relevantes. La disponibilidad de información estructurada y confiable brindó a los responsables administrativos una visión más clara del estado operativo del Corredor.

Gracias al sistema, fue posible identificar de manera más ágil situaciones como operadores con documentación vencida, registros incompletos o incidencias recurrentes, lo que permitió tomar acciones correctivas oportunas. Este acceso inmediato a la información redujo la dependencia de reportes manuales y mejoró la capacidad de respuesta ante situaciones operativas.

Asimismo, la consulta histórica de registros laborales y administrativos permite realizar análisis comparativos que antes resultaban difíciles de efectuar debido a la dispersión de la información. De esta manera, el sistema se convierte en una herramienta que no solo facilita la administración diaria, sino que también apoya la planificación y supervisión de las actividades del Corredor.

4.3 Resultados por módulos del sistema

Además de los beneficios generales obtenidos mediante la centralización de la información, cada uno de los módulos desarrollados aportó funcionalidades específicas que permitieron atender problemáticas particulares identificadas durante el análisis de los procesos administrativos del Corredor.

Módulo de operadores

La implementación del módulo de operadores permitió crear un expediente digital centralizado para cada empleado operativo. En este expediente se integró información personal, laboral y documental, incluyendo fotografías de credenciales, tarjetones, datos de contacto y fecha de ingreso.

Uno de los principales resultados obtenidos fue la disponibilidad inmediata de la información de los operadores desde cualquier equipo autorizado dentro de la organización. Asimismo, se logró mantener un historial actualizado de la documentación requerida para la operación del servicio.

Módulo de unidades

El módulo de unidades permitió concentrar en un solo repositorio digital la información relacionada con los autobuses que forman parte de la operación del Corredor.

La información registrada incluye placas, números económicos, características técnicas, kilometraje, vigencia de seguros y datos de aseguradoras. Gracias a esta integración fue posible disponer de información actualizada sobre cada unidad y relacionarla posteriormente con módulos como jornadas laborales, diésel, mantenimiento y siniestros.

Módulo de jornadas laborales

Este módulo permitió registrar la actividad operativa diaria de los operadores, asociando cada jornada con una unidad específica.

Como resultado, se logró mantener un historial detallado de entradas, salidas, tiempos de comida, desincorporaciones y observaciones operativas. La integración de esta información permitió establecer relaciones directas entre operadores, unidades y operación diaria.

Asimismo, este módulo se convirtió en la fuente principal de información utilizada posteriormente por Recursos Humanos para el seguimiento administrativo de las jornadas laborales.

Módulo de historial de jornadas

La principal aportación de este módulo fue la consolidación histórica de la información generada por las jornadas laborales.

La información puede consultarse mediante filtros por operador y periodos específicos, permitiendo revisar antecedentes operativos y verificar registros históricos sin necesidad de recurrir a documentación física.

Además, el sistema incorpora indicadores visuales que facilitan la interpretación de los registros almacenados.

Módulo de diésel

Este módulo permitió generar un historial digital de los suministros de combustible realizados a las unidades.

Cada carga queda asociada directamente al autobús correspondiente, permitiendo mantener un registro cronológico de abastecimientos y facilitando el análisis posterior del consumo de combustible.

La disponibilidad de esta información también permitió desarrollar herramientas de analítica orientadas al seguimiento operativo de cada unidad.

Módulo de siniestros

Antes del desarrollo del sistema no existía un expediente estructurado para documentar incidentes operativos.

Con la implementación de este módulo fue posible integrar en un mismo registro la información relacionada con el evento, incluyendo fotografías, ubicación geográfica, operador involucrado, unidad afectada, aseguradora y seguimiento administrativo.

La incorporación de coordenadas geográficas permitió además desarrollar herramientas de visualización y análisis espacial de los incidentes registrados.

Módulo de mantenimiento

La implementación de este módulo permitió generar historiales digitales de reparaciones e intervenciones realizadas a las unidades.

Cada registro almacena la descripción de la falla, las acciones correctivas realizadas y el personal responsable de la intervención, permitiendo mantener trazabilidad sobre el estado mecánico de la flota.

Asimismo, la información histórica facilita la identificación de fallas recurrentes y la planeación de actividades de mantenimiento.

Módulo de telemetría

El módulo de telemetría permitió incorporar herramientas de supervisión operativa en tiempo real.

Entre las principales funcionalidades implementadas se encuentran el monitoreo de jornadas activas, el control de tiempos de comida, la gestión de incidencias, el seguimiento de desincorporaciones y el registro inmediato de eventos operativos.

La integración de estos elementos permitió disponer de información actualizada sobre el estado de la operación en cada momento.

Gestión documental

La gestión documental permitió almacenar de manera digital fotografías, credenciales, tarjetones y diversos documentos administrativos asociados a operadores, unidades y procesos internos.

Este repositorio digital fortaleció la conservación de la información y facilitó la consulta inmediata de evidencias documentales desde los distintos módulos del sistema.

Integración de módulos

Uno de los resultados más importantes obtenidos fue la integración de los distintos módulos dentro de una misma plataforma. La información capturada en un módulo puede ser utilizada posteriormente por otros procesos del sistema gracias a las relaciones definidas en la base de datos.

Por ejemplo, un operador registrado en el módulo de Recursos Humanos puede ser asociado a jornadas laborales, recaudo, siniestros o incidencias operativas sin necesidad de capturar nuevamente su información. De manera similar, una unidad registrada puede

relacionarse con cargas de combustible, mantenimientos, jornadas laborales y eventos de siniestros.

Esta integración permitió construir una plataforma administrativa centralizada que proporciona trazabilidad sobre los procesos operativos y facilita el acceso a información relacionada entre las diferentes áreas del Corredor.

Área / Proceso	Método anterior	Tiempo promedio antes	Tiempo promedio actual	Reducción aproximada
Recursos Humanos – Consulta de expediente de operador	Búsqueda física en archivos y carpetas	15 a 30 min	10 a 20 seg	98%
Recursos Humanos – Consulta de vigencia de tarjetón	Revisión manual de expedientes	10 min a días	5 seg	99%
Operaciones – Consulta de jornada laboral	Búsqueda de hojas físicas en bodega o unidades	20 a 40 min	5 seg	99%
Recaudo – Consulta de ingresos por unidad	Revisión de cuadernos y hojas de cálculo	15 min	10 seg	98%
Diésel – Consulta de suministros por unidad	Revisión de formatos impresos	20 min	10 seg	99%
Siniestros – Localización de evidencia de accidente	Búsqueda en teléfonos celulares	Variable (hasta horas)	15 seg	>99 %
Electro-Mecánica – Consulta de historial de reparaciones	Revisión de cuadernos físicos	15 a 30 min	10 seg	98%
Telemetría – Identificación de operadores activos	Monitoreo manual y radio comunicación	10 min	Tiempo real	100%
Dirección – Obtención de información consolidada	Solicitud a múltiples áreas	Varias horas o días	Minutos	>95 %

Tabla 4.3 comprativa de tiempor por procesos antes y despues de la implementacion del sistema

4.4 Casos de aplicación y mejoras observadas

Durante el uso del sistema, se observaron casos de aplicación exitosos en la gestión administrativa del Corredor. Uno de los más representativos fue el módulo de registro y consulta de operadores, el cual permitió concentrar en un solo registro toda la información personal y evidencia documental requerida.

Como se muestra en la **Figura 4.2**, el sistema presenta en una sola vista los datos generales del operador, tales como número de empleado, nombre completo, teléfono, tipo de jornada y fecha de vencimiento del tarjetón. Esta visualización estructurada facilita la consulta inmediata de información que anteriormente se encontraba distribuida en expedientes físicos.

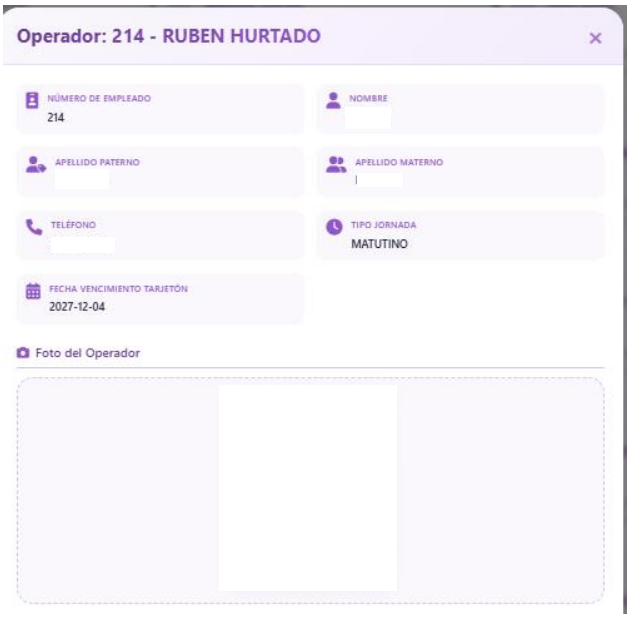

Figura 4.2. Datos del Operador

Asimismo, en la **Figura 4.3** se observa la incorporación de evidencia documental digital, incluyendo fotografía del operador, credencial oficial y tarjetón. La digitalización y almacenamiento en la base de datos de estos documentos eliminó la necesidad de consultar archivos físicos, reduciendo tiempos de búsqueda y riesgo de extravío de documentos.

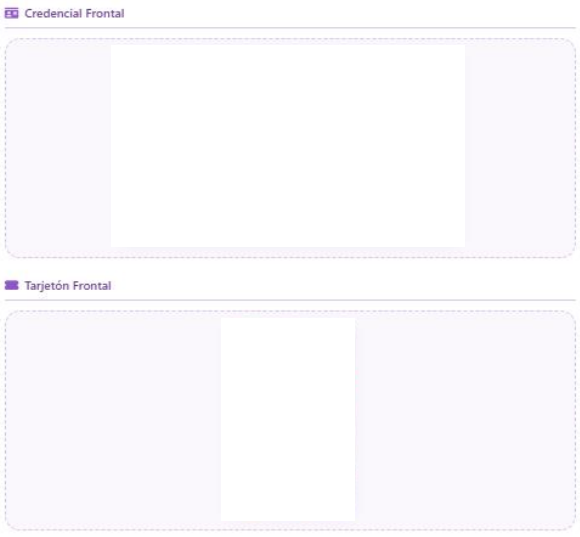

Figura 4.3. Evidencia documental

Este módulo permitió mejorar el control administrativo, ya que la información del operador se encuentra centralizada, organizada y respaldada digitalmente. Además, la consulta estructurada de registros facilitó la detección de inconsistencias, como documentos vencidos o datos incompletos, lo que anteriormente podía pasar desapercibido en revisiones manuales.

En conjunto, estas mejoras contribuyeron a una gestión más ordenada, con mayor trazabilidad de la información y una reducción significativa de errores derivados del manejo manual de expedientes.

4.5 Evaluación general de la solución

En términos generales, el sistema web desarrollado demostró ser una herramienta funcional y útil para la gestión administrativa del Corredor, aportando mejoras tangibles en eficiencia operativa, control de la información y apoyo a la toma de decisiones.

La implementación de la solución permitió modernizar procesos internos sin alterar de manera abrupta la operación diaria de la empresa, lo que facilitó su adopción por parte del personal administrativo. La disponibilidad de una plataforma centralizada contribuyó a mejorar la organización de la información y a reducir los tiempos asociados a tareas de registro, consulta y verificación de datos.

Los resultados obtenidos confirman que la implementación del sistema cumplió con los objetivos planteados, sentando las bases para una gestión administrativa más eficiente, organizada y orientada a la mejora continua.

Capítulo V. Conclusiones y trabajo futuro

Introducción

En este capítulo se presentan las conclusiones derivadas del desarrollo e implementación del sistema web de gestión administrativa para el Corredor, así como las líneas de trabajo a futuro que permitirían fortalecer y ampliar la solución propuesta. Este apartado reflexiona sobre el cumplimiento de los objetivos planteados, los beneficios obtenidos por la empresa y el impacto del proyecto tanto en el ámbito profesional como en el operativo.

5.1 Conclusiones

El desarrollo del sistema web de gestión administrativa permitió atender de manera efectiva la problemática identificada en el Corredor, relacionada con la falta de centralización, control y confiabilidad de la información administrativa. La sustitución de registros manuales y hojas de cálculo dispersas por una plataforma digital única contribuyó a mejorar la organización y disponibilidad de los datos.

La implementación del sistema integró procesos clave de la empresa, como el registro de operadores, la administración de unidades, el control de jornadas laborales y el seguimiento de incidencias, lo que permitió reducir errores derivados de la captura manual de información. Por ejemplo, el registro centralizado de operadores facilitó la consulta inmediata de datos y evidencias documentales, evitando la pérdida de información y reduciendo el tiempo necesario para su localización, lo que se tradujo en una mayor eficiencia operativa.

Asimismo, la disponibilidad de información estructurada y validada favoreció una mejor toma de decisiones administrativas, al permitir identificar de manera más ágil inconsistencias, registros incompletos o situaciones que requerían atención oportuna. Esto representó una mejora significativa respecto a los métodos previos, en los que la información se encontraba fragmentada y su consulta implicaba procesos manuales prolongados.

Desde el punto de vista tecnológico, el uso de herramientas como PostgreSQL, PHP, HTML, CSS y JavaScript demostró ser una alternativa viable y confiable para el desarrollo de soluciones empresariales. El sistema fue desplegado en un servidor web utilizando Internet Information Services (IIS), permitiendo la ejecución de la aplicación, la conexión con la base de datos PostgreSQL y el acceso remoto controlado mediante un navegador web. Esta arquitectura proporcionó un entorno estable para la administración centralizada de la información y la operación de los distintos módulos del sistema.

En el ámbito académico y profesional, el proyecto representó una oportunidad para aplicar de manera práctica los conocimientos adquiridos durante la formación como ingeniero en computación,

enfrentando un escenario real de análisis de requerimientos, diseño de base de datos, desarrollo incremental, pruebas, validación y despliegue de un sistema web en un entorno operativo.

En conjunto, los resultados obtenidos confirman que la solución tecnológica implementada cumplió con los objetivos establecidos y aportó beneficios tangibles a la gestión administrativa del Corredor.

5.2 Trabajo a futuro

A pesar de los resultados positivos alcanzados, el sistema desarrollado presenta oportunidades de mejora que pueden abordarse en etapas futuras. Una de las principales líneas de trabajo consiste en **ampliar la cobertura funcional del sistema**, incorporando nuevos módulos administrativos que fortalezcan el control interno y la operación del Corredor, tales como gestión avanzada de reportes, control de mantenimiento preventivo y seguimiento de indicadores clave de desempeño.

En materia de **seguridad y control de acceso**, se propone la adopción de ***frameworks* de desarrollo web** que integren de forma nativa mecanismos de autenticación robusta, cifrado de información y gestión avanzada de roles y permisos. *Frameworks* como ***Laravel*** incorporan componentes de seguridad que facilitan la implementación de autenticación basada en roles, protección contra ataques comunes (*CSRF, XSS, SQL Injection*) y manejo seguro de sesiones, lo que permitiría fortalecer la protección de la información sensible del sistema *(Laravel Documentation, 2024)*. Este tipo de herramientas reduce la implementación manual de controles de seguridad y se alinea con buenas prácticas recomendadas en la ingeniería de software *(OWASP, 2023)*.

Otra línea de mejora consiste en la **optimización del rendimiento y la escalabilidad del sistema**, con el objetivo de soportar un mayor volumen de información y usuarios concurrentes conforme la empresa continúe creciendo. Esto podría lograrse mediante la optimización de consultas a la base de datos, el uso de mecanismos de caché y una mejor gestión de recursos del servidor, prácticas recomendadas en sistemas web de uso empresarial (*PostgreSQL Global Development Group, 2024*).

Finalmente, el sistema puede evolucionar hacia una plataforma más integral mediante la incorporación de **herramientas de análisis y generación de reportes avanzados**, que permitan evaluar indicadores clave de desempeño y apoyar de manera más estratégica la toma de decisiones administrativas. La integración de estos mecanismos favorecería una gestión basada en datos, contribuyendo a la mejora continua de los procesos internos del Corredor y consolidando el sistema como una herramienta fundamental para la modernización del transporte público.

Referencias

1. Corredor Taxqueña Xochimilco Milpa Alta (COTAXOMIL). (s. f.). *Misión y visión*. https://cotaxomil.com/
2. Apache Friends. (2024). *XAMPP documentation*. https://www.apachefriends.org/about.html
3. Apache Software Foundation. (2024). *Apache HTTP Server documentation*. https://httpd.apache.org/docs/
4. Bass, L., Clements, P., & Kazman, R. (2013). *Software architecture in practice* (3rd ed.). Addison-Wesley.
5. Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of database systems* (7th ed.). Pearson Education.
6. IEEE Computer Society. (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK)* (Version 3.0). IEEE. https://www.computer.org/education/bodies-of-knowledge/software-engineering
7. Laudon, K. C., & Laudon, J. P. (2020). *Management information systems: Managing the digital firm* (16th ed.). Pearson Education.
8. Laravel LLC. (2024). *Laravel documentation*. https://laravel.com/docs
9. Mozilla Foundation. (2024). *MDN Web Docs: HTML*. https://developer.mozilla.org/en-US/docs/Web/HTML
10. Mozilla Foundation. (2024). *MDN Web Docs: CSS*. https://developer.mozilla.org/en-US/docs/Web/CSS
11. Mozilla Foundation. (2024). *MDN Web Docs: JavaScript*. https://developer.mozilla.org/es/docs/Web/JavaScript
12. OWASP Foundation. (2023). *OWASP Top Ten Web Application Security Risks*. https://owasp.org/www-project-top-ten/
13. PHP Group. (2024). *PHP documentation*. https://www.php.net/docs.php
14. PostgreSQL Global Development Group. (2024). *PostgreSQL documentation*. https://www.postgresql.org/docs/
15. Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill Education.
16. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database system concepts* (7th ed.). McGraw-Hill Education.
17. Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson Education.
18. Transportes Corredor Taxqueña–Milpa Alta. (s.f.). *Misión y visión de la empresa*. https://www.cotaxomil.com/
19. International Business Machines Corporation. (s.f.). *What is a DBMS?* https://www.ibm.com/topics/dbms
20. Mozilla Foundation. (s.f.). *How the web works*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works
21. Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGraw-Hill Education.
22. Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
23. World Wide Web Consortium (W3C). (s.f.). *HTML overview*. https://www.w3.org/TR/html52/