



Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

**Aplicación Coq's Tuning Performance, para
mejorar la Atención al Cliente mediante el
Procesado de Solicitudes en Segundo Plano**

Agosto 2021

Tesis presentada para obtener el título de
Ingeniería en Ciencias de la Computación

Presenta

Antonio Pérez Vázquez

Director

Mario Anzures García

Directora

María Luz Adolfina Sánchez Gálvez

Agradecimientos

A Dios por acompañarme y guiarme a lo largo de mi vida

A mis padres por el amor constante que me han brindado a lo largo de mi educación. Por ser un excelente ejemplo de vida a seguir. Gracias siempre.

A mis amigos, por sus aportaciones, vivencias, experiencias y consejos.

A mis asesores de tesis. Dr. Mario Anzures García y Dra. María Luz Adólfina Sánchez Gálvez, por brindarme tiempo, su dedicación, interés y consejos, por hacer esto posible. Gracias por todo.

Índice de contenido

1. INTRODUCCIÓN	1
1.1 PROBLEMA.....	2
1.2 OBJETIVOS	3
1.3 SOLUCIÓN.....	3
2. MARCO TEÓRICO	5
2.1 COMUNICACIÓN SÍNCRONA Y ASÍNCRONA	5
2.2 HTML	6
2.3 CSS.....	7
2.4 PHP.....	7
2.5 MYSQL	8
2.6 BOOTSTRAP	10
2.7 AJAX	10
2.8 SWEETALERT2.....	11
2.9 METODOLOGÍAS ÁGILES PARA EL DESARROLLO DE SOFTWARE	11
2.9.1 Scrum.....	12
2.9.2 Kanban.....	13
2.9.3 Metodología XP.....	14
2.10 METODOLOGÍA UTILIZADAS	15
3. ESTADO DEL ARTE	16
3.1 CHEVROLET	16
3.2 TOYOTA.....	16
3.3 HONDA	17
3.4 NISSAN.....	17
3.5 VOLKSWAGEN	17
4. DESARROLLO	18
4.1 PÁGINA DE INICIO.....	22
4.2 MANEJO DE SESIONES DE USUARIO	23
4.3 GESTIÓN DE CITAS.....	24
4.3.1 Eliminación de una Cita.....	27
4.3.2 Aceptación de una Cita	28
4.3.3 Alerta Push.....	30
4.3.4 Cancelación de una Cita	31
4.3.5 Asignación de un Técnico	33
4.3.6 Envío de Correos	35
4.4 CHAT CON AJAX Y PHP	39
4.4.1 Maquetado del Chat.....	40
4.4.2 Estatus Typing	42
4.4.3 Manejo de Mensajes	43
4.4.4 Mensajes sin Leer	46
4.5 ADMINISTRADOR DE PERSONAL	49

4.5.1 Despliegue de Registro de Personal.....	50
4.5.2 Registro de Personal.....	51
4.5.3 Actualización de Personal.....	52
4.5.4 Eliminación de Personal.....	54
5. CONCLUSIONES Y TRABAJO FUTURO	57
BIBLIOGRAFÍA.	58

Índice de Figuras.

FIGURA 1. DIAGRAMA DE BLOQUES	19
FIGURA 2. ÍNDEX DEL SISTEMA COQ'S TUNING PERFORMANCE..	22
FIGURA 3. INICIO DE SESIÓN AL SISTEMA..	23
FIGURA 4. MENÚ DEL USUARIO CLIENTE.	24
FIGURA 5. MANEJO DE CITAS POR PARTE DEL ADMINISTRADOR.	25
FIGURA 6. FUNCIÓN MOSTRAR TABLA.....	25
FIGURA 7. DOCUMENTO QUE LISTA LAS CITAS.....	26
FIGURA 8. TEMPLATE DE CITAS	26
FIGURA 9. FUNCIÓN ELIMINAR	27
FIGURA 10. CONSULTA SQL ELIMINAR.....	28
FIGURA 11. FUNCIÓN ACEPTAR	28
FIGURA 12. ESTRUCTURA DE LAS ALERTAS DE CONFIRMACIÓN.....	29
FIGURA 13. CONSULTA SQL ACTUALIZAR.....	30
FIGURA 14. FUNCIÓN ALERTA.	30
FIGURA 15. VISUALIZACIÓN DE LA NOTIFICACIÓN TIPO PUSH.	31
FIGURA 16. FUNCIÓN CANCELAR.	32
FIGURA 17. QUERY DE CANCELACIÓN	32
FIGURA 18. ASIGNACIÓN DE EMPLEADO	33
FIGURA 19. FUNCIÓN ASIGNAR MECÁNICO	34
FIGURA 20. LECTURA DE LA VARIABLE EDIT	34
FIGURA 21. QUERY QUE ASIGNA UN MECÁNICO.....	35
FIGURA 22. FORMULARIO DE DETALLES DEL SERVICIO.....	36
FIGURA 23. CAMPOS HIDDEN DEL FORMULARIO.....	36
FIGURA 24. FUNCIÓN DATOS DE LA CITA	37
FIGURA 25. EXTRACCIÓN DE LOS DATOS DE LA CITA	37
FIGURA 26. IMPRESIÓN EN CONSOLA DE LOS DATOS DE LA BD	38
FIGURA 27. ESTRUCTURA DEL MENSAJE.....	39
FIGURA 28. VISTA DEL CHAT ENTRE USUARIO Y ADMINISTRADOR	40
FIGURA 29. VISTA DEL CHAT ENTRE ADMINISTRADOR Y USUARIOS	41
FIGURA 30. FUNCIÓN UPDATEUSERLIST	42
FIGURA 31. USUARIO EN LÍNEA Y DESCONECTADO	42
FIGURA 32. FUNCIÓN DE ESTATUS DE ESCRITURA	43
FIGURA 33. VISUALIZACIÓN DEL ESTATUS DE ESCRITURA.....	43
FIGURA 34. FUNCIÓN SHOWUSERCHAT	44
FIGURA 35. CONDICIONAL SHOW_CHAT	44

FIGURA 36. SEGUNDA FUNCIÓN SHOWUSERCHAT.....	45
FIGURA 37. FUNCIÓN GETUSERCHAT	46
FIGURA 38. FUNCIÓN UPDATEUNREADMESSAGECOUNT.....	47
FIGURA 39. CONDICIONAL UPDATEUNREADMESSAGECOUNT.....	47
FIGURA 40. FUNCIÓN GETUNREADMESSAGECOUNT	48
FIGURA 41. VISTA GENERAL DE UN CHAT CON MENSAJES SIN LEER	48
FIGURA 42. PESTAÑA DEL ADMINISTRADOR PARA GESTIONAR PERSONAL	49
FIGURA 43. PRIMER FRAGMENTO DE LA FUNCIÓN MOSTRARTABLA().....	50
FIGURA 44. CONSULTA DE PERSONAL EN LA BASE DE DATOS	50
FIGURA 45. SEGUNDO FRAGMENTO DE LA FUNCIÓN MOSTRAR TABLA.....	51
FIGURA 46. FUNCIÓN CONDICIONAL	51
FIGURA 47. INSERCIÓN EN LA BASE DE DATOS DE NUEVO PERSONAL.....	52
FIGURA 48. FUNCIÓN QUE TRAE A PANTALLA LOS DATOS	53
FIGURA 49. MECÁNICO-SIMPLE.PHP.....	53
FIGURA 50. MECÁNICO-EDIT.PHP	54
FIGURA 51. FUNCIÓN ELIMINAR PARTE UNO	55
FIGURA 52. QUERY DE ELIMINACIÓN	55
FIGURA 53. FUNCIÓN ELIMINAR PARTE DOS	56
FIGURA 54. SWEETALERT2	56

Índice de tablas

TABLA 1. APLICACIONES SIMILARES	16
---------------------------------------	----

1. Introducción

Los automóviles han estado presentes por cerca de 150 años, cuando el 6 de octubre de 1866, circuló en Estados Unidos el primer automóvil movido a vapor. Desde entonces surgió la necesidad de brindar servicio a estas unidades, así como también, este nuevo invento pasó de ser un lujo a una necesidad.

Es aquí donde nace el servicio automotriz, para ofrecer a los dueños de los automóviles una asistencia preventiva y correctiva a los motores, los cuales, al ser impulsados por combustión interna, generan calor; y como es bien conocido, el enemigo de todo mecanismo motriz es el calor, que a su vez genera desgaste en sus componentes.

Estos servicios han ido cambiando a lo largo del tiempo, y han evolucionado a la par de los motores. Desde la implementación de aditivos que ayuden a mejorar el desempeño del automóvil, hasta motores más eficientes y económicos en cuanto a gasto de combustible se refiere.

Por otro lado, los servicios se han automatizado y vuelto más ágiles. Años atrás se tenían que hacer tediosas filas de espera para realizar algún trámite o servicio, lo cual hacía que se tuviera que destinar más tiempo al tener que llevarlo a cabo.

Si se realiza un análisis en cuanto a servicios automotrices en talleres, ninguno de ellos cuenta con una plataforma o aplicación para gestionar solicitudes. Solo las agencias de las principales marcas en el mercado han empezado a implementarlas. Especialmente en tiempos de contingencia sanitaria han ido incrementando estas herramientas.

Las aplicaciones web han ido evolucionando al paso de los años también y cada vez brindan mayores beneficios. En un principio solo eran informativas, después se empezaron a hacer comunes los blogs, donde los usuarios podían compartir experiencias o información. Y finalmente se tienen aplicaciones complejas que incluso no es necesario actualizar para recibir información.

1.1. Problema

En la actualidad, se tienen diferentes opciones al momento de elegir un servicio para nuestra unidad o automóvil. Por un lado, existe la opción de llevar el vehículo a una agencia certificada; donde se ofrece un buen servicio, pero a un alto costo. Recalcando que los tiempos de entrega del vehículo son prolongados y existen algunas agencias que no respetan los tiempos que ellos establecen al realizar un trabajo.

Algunos necesidades y problemas que se presentan actualmente al requerir un servicio automotriz son:

- Los elevados precios de un servicio preventivo o correctivo.
- El alto costo de combustible
- La eficiencia en el consumo de combustible
- La falta de seguridad en los vehículos
- El desempeño del motor
- La confianza entre el cliente y la agencia automotriz
- La forma de realizar los servicios (en cuanto a agenda y tramite se refiere).

Hoy en día el mundo se ha reinventado y automatizado en gran medida. Se tienen aplicaciones y plataformas que brindan diversos servicios, facilitando la tarea, evitando filas y tediosos tiempos de espera. Por tanto, en este trabajo de tesis se plantea el desarrollo de una aplicación Web para un taller mecánico con la finalidad de brindar una herramienta a los dueños de dicho taller (pequeño o mediano) con la principal función de establecer una comunicación más efectiva entre el cliente y el taller. De esa manera, se incide en el modelo de negocio de este tipo de pequeñas o medianas empresas; teniendo como finalidad cubrir las necesidades de un mercado creciente y súper desarrollado como lo es actualmente la industria automotriz. La mayoría de las veces el tiempo con el que se cuenta es limitado, y como tal, es un recurso de mucho valor, considerado un lujo, con el que la mayoría de las veces no se cuenta.

1.2. Objetivos

El objetivo general del proyecto de tesis es: Utilizar las técnicas de AJAX en la gestión de servicios en una aplicación para un taller mecánico; mejorando la relación entre el cliente y el taller, optimizando la respuesta y experiencia del usuario para ir aumentando el número de clientes. Los objetivos específicos se centran en:

- Aplicar AJAX en la gestión de servicios para reducir tiempos de respuesta de cada petición.
- Hacer uso de formularios para obtener información de los servicios realizados
- Mantener comunicación con el cliente con respecto al estado de la unidad y detalles del servicio en proceso
- Enviar notificaciones con respecto al próximo servicio a realizar
- Tener un buzón de dudas y preguntas frecuentes para que los clientes y futuros clientes puedan resolver
- Brindar al cliente una interfaz usable, robusta, responsiva, segura y con un alto desempeño y tiempo de respuesta corto.
- Detallar los servicios y costos que se ofrece a nuestros clientes
- Informar a los clientes acerca de las nuevas ofertas y promociones de cada mes

1.3. Solución

Por ende, es necesario contar con una aplicación que permita simplificar el proceso que se tiene que realizar en un servicio automotriz, tanto para el cliente como para el prestador de tal servicio. En este trabajo, se piensa que una aplicación Web es la ideal, porque la interacción se puede realizar desde donde se encuentre tanto el cliente como el prestador del servicio automotriz, reduciendo de esta manera esfuerzo, tiempo y gastos para ambas partes.

La aplicación Web de Coq's Tuning Performance, para mejorar la atención al cliente, ofrece al usuario la capacidad de agendar una cita vía web, donde se elige el horario deseado o a disponibilidad dependiendo la demanda. Con esta herramienta de software web se pretende reducir tiempos de espera por parte de los usuarios en cuanto a reparaciones y servicios, además que se puede administrar mejor el tiempo u horario del taller. Así como ofrecer a los usuarios la oportunidad de recibir un servicio de excelente calidad, con el mejor costo, dando más valor por su dinero. En consecuencia, se evita a los usuarios la molestia de salir de casa u oficina para ir a entregar su unidad o bien para recogerla del taller, teniendo que perder varias horas.

En consecuencia, en este trabajo de tesis se plantea la aplicación del conjunto de técnicas AJAX en la gestión de citas de una plataforma de servicios automotrices con la finalidad de brindar una herramienta a los dueños de dicho taller (pequeño o mediano) con la principal función de optimizar en el backend las peticiones al servidor y hacer que la comunicación sea mucho más rápida y efectiva sin tener que actualizar el panel. Y del mismo modo para el funcionamiento del chat entre clientes y administrador, en el que se crea un canal de comunicación más personalizada y efectiva.

El sistema será desarrollado utilizando Sublime text [1], Brackets [2], XAMPP [3], Bootstrap [4], HTML [5], CSS [6], PHP [7], MySQL [7], 000Webhost [8], AJAX [9] y SweetAlert2 [20].

El documento de tesis se organiza de la siguiente forma: Sección 2 presenta el marco teórico que sustenta el sistema Web de Coq's Tuning Performance. Sección 3 expone el estado del arte de este trabajo de tesis. Sección 4 explica el desarrollo del sistema Web de Coq's Tuning Performance. Sección 5 describe las conclusiones y el trabajo futuro.

2. Marco Teórico

La web en la actualidad es una potente herramienta como espacio de información que día a día exige innovadoras funcionalidades para cubrir las complejas exigencias de los usuarios. El desarrollo web es una parte fundamental para poder cubrir estas exigencias. Las herramientas que se utilizan para crear estos complejos sitios cada día evolucionan para dar paso a nuevas tecnologías de desarrollo más potentes, robustas y que incluso se pueden adaptar a un proyecto web específico.

La base de todo proyecto web en cuanto a infraestructura es HTML5 este te provee los elementos estructurales y de maquinación que también te permite incluir otro tipo de tecnología como son las hojas de estilo en cascada CSS3 que te ayuda a dar diseño a la estructura HTML esto para que la web tenga una mejor apariencia visual ante el usuario. Para que la página no sea simple texto plano, tenga interacción con el usuario y sea dinámica se usa PHP.

En conjunto con todas estas tecnologías mencionadas se pueden desarrollar páginas web informativas estas siempre se van a ejecutar en el navegador del usuario debido a esto dichas paginas estarán limitadas en cuanto a funcionalidades. Por otro lado, existen páginas más complejas que necesitan ejecutarse en un servidor, para esto se usan lenguajes del lado del servidor como PHP. Este tipo de sitios ven la necesidad de incluir bases de datos para que el usuario pueda almacenar y visualizar información, para esto se usan gestores base de datos tales como MYSQL.

2.1. Comunicación síncrona y asíncrona

En los inicios de Internet las comunicaciones se basaban en el modelo cliente – servidor en el que el usuario realizaba una consulta (cliente) y el servidor respondía a esta petición. Este comportamiento se conoce como comunicación síncrona y, por tanto, a aquellas aplicaciones que siguen este modelo se les denomina aplicaciones síncronas.

Por otro lado, las aplicaciones asíncronas permiten la notificación de un cliente cuando un evento ocurre en la parte del servidor. De esta forma, las aplicaciones asíncronas pueden actualizar su información más eficiente y rápidamente que las síncronas. Para permitir un comportamiento asíncrono la comunicación se produce en paralelo (simultáneamente) con el flujo síncrono (principal).

Es por ello, que en este trabajo de tesis se plantea la aplicación del conjunto de técnicas AJAX en la gestión de citas de una plataforma de servicios automotrices con la finalidad de brindar una herramienta a los dueños de dicho taller con la principal función de optimizar en el backend las peticiones al servidor y hacer que la comunicación sea mucho más rápida y efectiva sin tener que actualizar el panel.

Además, se hace uso de Brackets y sublime Text para editar los archivos de la aplicación, así como PHP, Bootstrap para desarrollar la aplicación con plantillas responsivas intuitivas, usables y SweetAlert2 para generar alertas agradables y con ligeras animaciones que salen de lo común. También se hará uso de Webhost para la implantación del sistema Web de Coq's Tuning.

2.2. HTML

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5, es simplemente un conjunto de nuevas funciones disponibles para el desarrollo de aplicaciones web, que se agregan a las capacidades existentes que se encuentran en HTML4. Está especialmente diseñado para mejorar el lenguaje con un soporte mucho mejor para la comunicación multimedia y del servidor, lo que facilita mucho el trabajo de un desarrollador web [10].

HTML5 no es una nueva versión de HTML4 en comparación con el lanzamiento de nuevas versiones de software. Comprende un conjunto completo

de pequeñas adiciones al estándar web existente. Actualmente, cada navegador implementa algunas de estas características, pero no todas.

2.3. CSS

CSS es un lenguaje que trabaja junto con HTML5 para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, etc. En un intento por reducir el uso de código Javascript y para estandarizar funciones populares fue que se creó esta tecnología. No solo cubre diseño y estilos web sino también forma y movimiento. La especificación de CSS3 es presentada en módulos que permiten a la tecnología proveer una especificación estándar por cada aspecto involucrado en la presentación visual del documento.

La especificación de HTML5 fue desarrollada considerando CSS a cargo del diseño. Debido a esta consideración, la integración entre HTML y CSS es ahora vital para el desarrollo web y esta es la razón por la que cada vez que se menciona HTML5 también se hace referencia a CSS3, aunque oficialmente se trate de dos tecnologías completamente separadas [11, 12].

2.4. PHP

PHP es un lenguaje de propósito general y de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML. Su sintaxis recurre a C, Java y Perl, siendo así sencillo de aprender. El objetivo principal de este lenguaje es permitir a los desarrolladores escribir dinámica y rápidamente páginas web generadas [13].

PHP es un lenguaje interpretado del lado del servidor. Se caracteriza por su potencia, versatilidad, robustez y modularidad. PHP es multiplataforma, esto quiere decir, que es capaz de funcionar en más de una arquitectura de ordenador o sistema operativo, tal como Linux, muchas variantes Unix (incluido HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS, etc. PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, *Microsoft Internet Information Server*, *Personal Web Server*, *Netscape* y *iPlanet*, *Oreilly*

Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI. Esto puede ser una tarea que consume tiempo, ya que los diferentes sistemas operativos tienen diferentes interfaces de programación de aplicaciones o API [13].

En definitiva, PHP es uno de los lenguajes más utilizados actualmente en el desarrollo de aplicaciones web y viene experimentando un constante crecimiento en su nivel de utilización en Internet.

2.5. MYSQL

MySQL es un sistema de gestión de bases de datos. Una base de datos es una colección estructurada de datos. Puede ser cualquier cosa, desde una simple lista de compras hasta una galería de imágenes o la gran cantidad de información en una red corporativa. Para agregar, acceder y procesar datos almacenados en una base de datos, necesita un sistema de administración de base de datos como MySQL Server. Dado que las computadoras son muy buenas para manejar grandes cantidades de datos, los sistemas de administración de bases de datos desempeñan un papel central en la informática, como utilidades independientes o como parte de otras aplicaciones.

Las bases de datos MySQL son relacionales. Una base de datos relacional almacena los datos en tablas separadas en lugar de colocar todos los datos en un gran almacén. Las estructuras de la base de datos están organizadas en archivos físicos optimizados para la velocidad. El modelo lógico, con objetos como bases de datos, tablas, vistas, filas y columnas, ofrece un entorno de programación flexible. Usted configura reglas que gobiernan las relaciones entre los diferentes campos de datos, como uno a uno, uno a muchos, únicos, requeridos u opcionales, y punteros entre diferentes tablas. La base de datos hace cumplir estas reglas, de modo que, con una base de datos bien diseñada, su aplicación nunca ve datos inconsistentes, duplicados, huérfanos, desactualizados o faltantes [14].

El software MySQL es Open Source. Se puede estudiar el código fuente y cambiarlo para adaptarlo a sus necesidades. El software MySQL utiliza la GPL (licencia pública general de GNU), para definir lo que puede y no puede hacer con el software en diferentes situaciones. Si no se siente cómodo con la GPL o necesita integrar código MySQL en una aplicación comercial, puede comprar una versión con licencia comercial.

El servidor de base de datos MySQL es muy rápido, confiable, escalable y fácil de usar. MySQL Server puede ejecutarse cómodamente en una computadora de escritorio o portátil, junto con otras aplicaciones, servidores web, etc., que requieren poca o ninguna atención. Si dedica una máquina completa a MySQL, puede ajustar la configuración para aprovechar toda la memoria, la potencia del CPU y la capacidad de E / S disponible. MySQL también puede escalar a grupos de máquinas conectadas en red.

MySQL Server se desarrolló originalmente para manejar grandes bases de datos mucho más rápido que las soluciones existentes y se ha utilizado con éxito en entornos de producción altamente exigentes durante varios años. Aunque está en constante desarrollo. Su conectividad, velocidad y seguridad hacen que MySQL Server sea muy adecuado para acceder a bases de datos en Internet.

MySQL Server funciona en cliente / servidor o sistemas embebidos. El software de base de datos MySQL es un sistema cliente / servidor que consta de un servidor SQL multiproceso que admite diferentes back-end, varios programas y bibliotecas diferentes para el cliente, herramientas administrativas y una amplia gama de interfaces de programación de aplicaciones (API) [14].

También MySQL Server puede funcionar como una biblioteca multiproceso incrustada para vincular la aplicación y obtener un producto independiente más pequeño, más rápido y más fácil de administrar. MySQL Server tiene un conjunto práctico de características desarrolladas en estrecha colaboración con sus usuarios.

2.6. Bootstrap

Bootstrap fue creado por Mark Otto y Jacob Thornton en Twitter como un marco para fomentar la coherencia entre las herramientas internas. Ahora es un proyecto de código abierto alojado en GitHub y ha visto un rápido crecimiento y uso global en aplicaciones web y sitios web.

Cuando Bootstrap se creó por primera vez en Twitter, se creó como un conjunto de herramientas reutilizables, componentes con documentación adicional y fragmentos de código sobre cómo usarlos. Esto ayudó a un equipo de múltiples desarrolladores a trabajar en un proyecto y tener una cohesión metodológica sobre cómo construir diseños [15].

2.7. AJAX

AJAX es un conjunto de técnicas de desarrollo web que permiten que las aplicaciones web funcionen de forma asíncrona, procesando cualquier solicitud al servidor en segundo plano. Significa JavaScript asíncrono y XML (Asynchronous JavaScript and XML).

JavaScript es un lenguaje de programación muy conocido. Entre otras funciones, gestiona el contenido dinámico de un sitio web y permite la interacción dinámica del usuario. XML es otra variante de un lenguaje de marcado como HTML, como lo sugiere su nombre: eXtensible Markup Language. Mientras HTML está diseñado para mostrar datos, XML está diseñado para contener y transportar datos.

Tanto JavaScript como XML funcionan de forma asíncrona en AJAX. Como resultado, cualquier aplicación web que use AJAX puede enviar y recuperar datos del servidor sin la necesidad de volver a cargar toda la página.

AJAX se ha utilizado en la presente tesis como una técnica para desarrollar aplicaciones dinámicas que ayuden a que el usuario no permanezca largos ratos esperando por el retorno de una página, pues esta técnica hace que

rutinas y scripts viajen al servidor en busca de datos para actualizar ciertas partes y así mostrar u ocultar ciertas porciones [10].

2.8. SweetAlert2

Hay muchas formas de lanzar notificaciones o alertas en una web. Una de esas formas es usando la función `alert` de JavaScript, que muestra una caja de alerta con un mensaje que se desee. En los tiempos que corren esta es una opción poco estética, con SweetAlert2 se consigue dar a los usuarios notificaciones y alertas de un modo mucho más visual.

Con este plugin jQuery se da un aspecto a los mensajes que se envían a los usuarios acorde a las tendencias actuales. Además, se tiene la posibilidad de configurar el plugin de muchas formas diferentes.

Siempre es importante dar respuestas a las acciones realizadas por el usuario y hay muchas formas de lanzar notificaciones o alertas en una web o sistema web. La idea es dejar de usar el famoso `alert()` de javascript que, aunque funciona no es estético por su diseño y animación antigua... Así que para esta solución se cuenta con SweetAlert2, con ella se consigue dar notificaciones y alertas hermosas, estéticas y mucho más funcionales. Con esto se busca reemplazar el aburrido mensaje de `alert` o `confirm` nativo de Javascript [20].

2.9. Metodologías ágiles para el desarrollo de software

Para el desarrollo de proyectos de software (en este caso de proyectos web) existen diversas metodologías que ofrecen diferentes ventajas y trabajan de diferente forma de acuerdo al tipo de proyecto a realizar.

Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del

ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo.

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.

Las Metodologías Ágiles son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.

En esencia, las empresas que apuestan por esta metodología consiguen gestionar sus proyectos de forma eficaz reduciendo los costes e incrementando su productividad. Este tipo de Metodologías son aquellas que básicamente se basan en interactuar directamente con el cliente desarrollando y entregando avances parciales del trabajo hasta tener la versión final.

2.9.1 Scrum

Scrum es una metodología Ágil y flexible que trata de ser de utilizar en proyectos donde el nivel de incertidumbre es alto. Su objetivo será controlar y planificar proyectos con un gran volumen de cambios de última hora, en donde la incertidumbre sea elevada.

La metodología Scrum se centra en ajustar sus resultados y responder a las exigencias reales y exactas del cliente. De ahí, que se vaya revisando cada entregable, ya que los requerimientos van variando a corto plazo.

Entre las principales características de la metodología Scrum, desataca que es un desarrollo incremental en lugar de la clásica planificación del desarrollo

completo de un producto o servicio. Sus equipos de trabajo se caracterizan por ser auto-organizados. Y se centra en el producto final, en la calidad del mismo.

Para poder implementar Scrum es necesario saber que esta se basa básicamente en hacer reuniones periódicas con base a ver los avances del proyecto, es decir se realizan reuniones para asegurar el cumplimiento de los objetivos establecidos, a este tipo de reuniones en este modelo se le conocen como Sprints, estos son el corazón de Scrum, es el contenedor de los demás eventos. Todo lo que ocurre para entregar valor está dentro de un Sprint. La duración máxima es de 1 mes, el tiempo se determina en base al nivel de comunicación que el cliente quiere tener.

De esta manera se puede tener un desarrollo iterativo pues los Sprints deben de llevarse a cabo periódicamente. En pocas palabras un Sprint es un conjunto de reuniones que se realizan durante el desarrollo del proyecto.

2.9.2 Kanban

Actualmente, el término Kanban ha pasado a formar parte de las llamadas metodologías ágiles, cuyo objetivo es gestionar de manera general cómo se van completando las tareas. Kanban es una palabra japonesa que significa tarjetas visuales, donde Kan es visual, y Ban corresponde a tarjeta.

Las principales ventajas de esta metodología es que es muy fácil de utilizar, actualizar y asumir por parte del equipo. Además, destaca por ser una técnica de gestión de las tareas muy visual, que permite ver a golpe de vista el estado de los proyectos, así como también pautar el desarrollo del trabajo de manera efectiva.

Para ello, simplemente se debe crear un tablero, que deberá ser visible y accesible por parte de todos los miembros del equipo. Cada una de las columnas corresponderá a un estado concreto del flujo de tareas, que sirven para saber en qué situación se encuentra cada proyecto. El tablero debe tener tantas columnas como estados por los que pasa una tarea, desde que se inicia

hasta que finaliza (p.e: diagnóstico, definición, programación, ejecución, testing, etc.).

2.9.3 Metodología XP

Extreme Programming o Programación Extrema es una de las llamadas metodologías Ágiles de desarrollo de software más exitosas.

La programación extrema o eXtreme Programming (XP) es un enfoque de la ingeniería de software formulado por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

El modelo XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. El método especifica que, de estas

cuatro variables, tres de ellas podrán ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto), y el valor de la restante deberá ser establecida por el equipo de desarrollo, quien establecerá su valor en función de las otras tres.

Por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que durará el proyecto. Se trata de establecer un equilibrio entre las cuatro variables del proyecto.

2.10 Metodología utilizada

En este proyecto de tesis se trabajó con la metodología XP, ya que, por los tiempos y requerimientos del proyecto, es la que mejor se adapta. Y, por otro lado, en proyectos anteriores de Ingeniería Web he trabajado con esta metodología, su ciclo de vida o iteraciones ayudan bastante a ir generando pequeñas versiones funcionales que se van integrando al software una vez que han sido concluidas y aprobadas.

En lo personal la Programación Extrema me hace trabajar de una manera más organizada y con una menor tasa de errores, además que al estar en contacto cercano con el cliente se realiza un proyecto funcional, que puede adaptarse a los requerimientos que pueden surgir sobre la marcha y realizarse las adaptaciones correspondientes.

3 Estado del Arte

En cuanto a aplicaciones similares se puede observar que ningún taller de servicio particular cuenta con una aplicación web que agende citas en línea, y su principal problema es que a veces el trabajo se junta en exceso y los usuarios son los que se ven afectados al tener que esperar más de la cuenta.

Las agencias automotrices han empezado a implementar esta herramienta, para poder beneficiar tanto a los clientes que reciben día con día en el taller, como también a ellos mismos para distribuir sus tiempos; optimizando sus recursos en cuanto al número de trabajadores que requieren, para poder brindar un servicio más eficiente.

Tabla 1. Aplicaciones similares.

<i>Empresa</i>	<i>Servicio</i>	<i>Descripción</i>
<i>Chevrolet</i>	Citas en línea Buzón de comentarios	En esta agencia este sistema de citas en línea lleva aproximadamente 2 años de implementación y en lo personal si hace una diferencia con respecto a los tiempos de espera. Ya que si se llega sin cita el tiempo de servicio puede variar [16]. Por otro lado, se tiene un excelente servicio postventa, ya que la empresa se comunica con el cliente vía telefónica para hacer un breve cuestionario con respecto al servicio y calificar los aspectos más generales
<i>Toyota</i>	Citas en línea Buzón de comentarios	En esta agencia este sistema de citas en línea está en proceso de implementación. Además que cuentan con un buzón de recomendaciones que ayuda a retroalimentar a la empresa sobre la satisfacción del cliente [17]

Honda	Citas en línea Buzón de comentarios	En esta agencia este sistema de citas en línea lleva aproximadamente 1 año de implementación para optimizar los tiempos de espera. Ya que si se llega sin cita el tiempo de servicio puede variar [18]. Cuenta con servicio post venta, pero no suele ser muy eficiente.
Nissan	Citas en línea Buzón de comentarios	En su sitio web se encuentra en la pestaña de servicio un formulario donde incluso se localiza al distribuidor o agencia de servicio más cercana. Y elegir la fecha del servicio [19] No cuenta con un Buzón de opiniones. Sin embargo, cuenta con un Bot para despejar dudas
Volkswagen	Citas en línea Servicio a domicilio	Esta agencia no contaba con citas en línea. Pero a raíz de la pandemia implementó muchos servicios a domicilio [20]. Sus citas en línea aún se encuentran en prototipo, ya que solo se puede buscar la concesionaria más cercana y de ahí ya no manda más.

4 Desarrollo

Para la elaboración de este proyecto se utilizó el conjunto de técnicas y lenguajes antes mencionadas en el marco teórico, con la finalidad de desarrollar un sitio web responsivo, ergonómico, seguro, robusto y usable.

Como bien se sabe, las aplicaciones síncronas presentan una serie de limitaciones que, con el paso del tiempo y la evolución del uso y la capacidad de la red, han terminado por ser determinantes en el desarrollo de tecnologías alternativas. Una de las principales limitaciones involucra la actualización de los datos que se muestran en pantalla. La aplicación debe ser capaz de actualizar los valores en un determinado intervalo de tiempo. El problema, obvio, que surge de este modo de funcionamiento es que existirá un lapso de tiempo en el que los datos estén desactualizados y esto puede suponer una merma importante de las prestaciones de una aplicación.

En estos casos y para superar estas limitaciones, las aplicaciones síncronas están siendo sustituidas por lo que se conoce por aplicaciones asíncronas. Las cuales permiten la notificación de un cliente cuando un evento ocurre en la parte del servidor. De esta forma, las aplicaciones asíncronas pueden actualizar su información de forma mucho más eficiente y rápida que las síncronas. Este comportamiento asíncrono de la comunicación se produce en paralelo (simultáneamente) con el flujo síncrono (principal).

Con el uso de AJAX, se logra que el usuario se relacione con las aplicaciones de una forma mucho más natural, sin tener que actualizar una pestaña del sitio. Conduciendo a desarrollar soluciones que se integren mejor y tengan una respuesta más eficiente en determinados entornos, donde el usuario necesita ver que los cambios son actualizados en todo momento y se ven inmediatamente reflejados en la aplicación. Ya que la información es un bien valioso y la rapidez de respuesta una virtud necesaria.

El sistema web Coq's Tuning Performance es una herramienta que ayuda a gestionar citas de servicio, el personal y los servicios que se ofrecen en un taller mecánico automotriz. Por medio de este sistema los visitantes del sitio pueden visualizar datos de interés con respecto al cuidado del vehículo, los

servicios que se ofrecen con sus respectivos costos, las ofertas, promociones o paquetes del mes, un mapa de Google que indica la ubicación del taller para que se les facilite llegar. Además de contar con un apartado para registrarse o iniciar sesión y; una vez que accedan, poder agendar una cita, verificar su estatus e incluso mandar mensaje en un chat privado entre el cliente y el administrador.

En la figura 1 se presenta un panorama general de la aplicación que se establece por medio de un diagrama de bloques, donde se encuentran los diferentes módulos de esta aplicación.

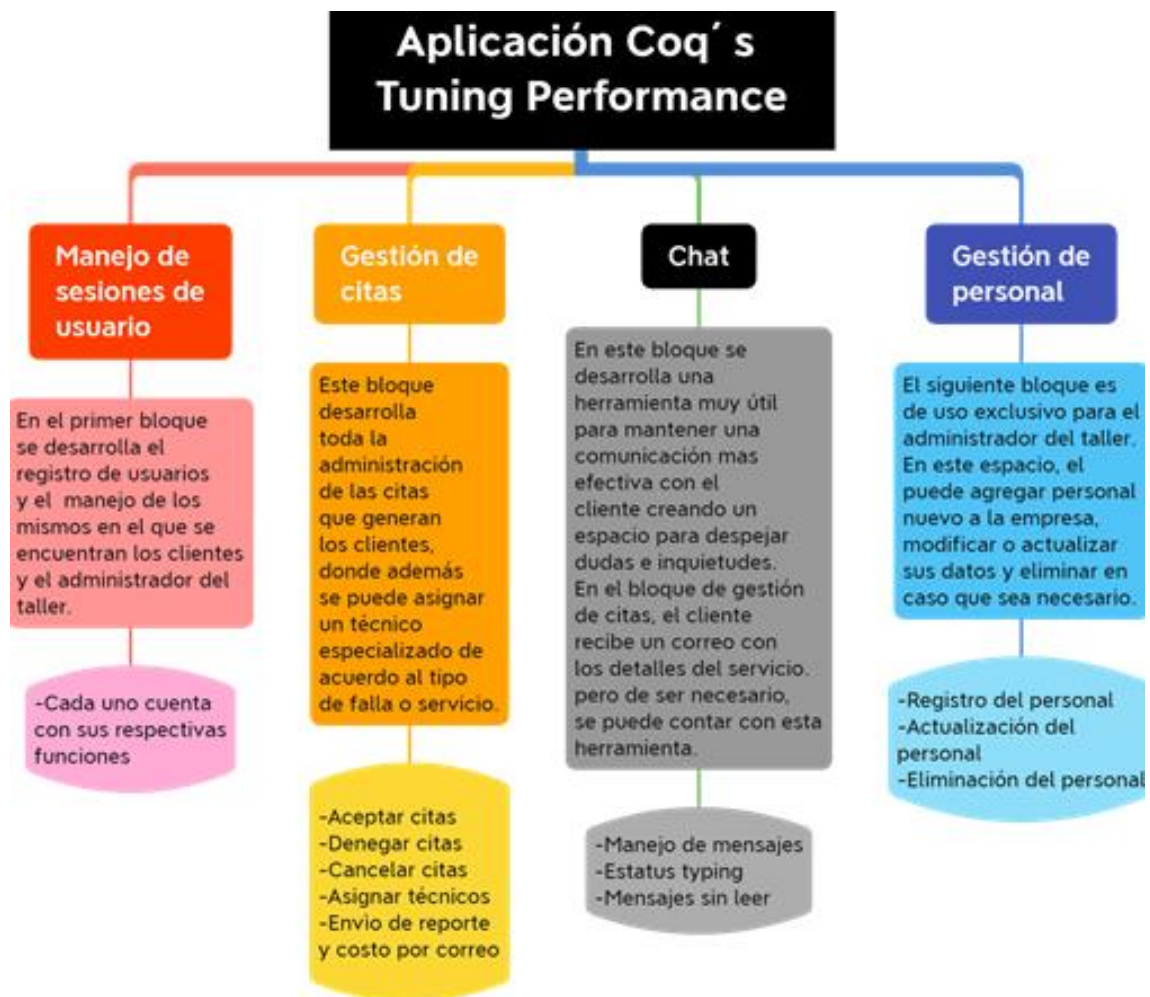


Figura 1. Diagrama de bloques

En el primer bloque está el manejo de sesiones de usuario, en el que se realiza el registro y posteriormente el ingreso al sistema, donde cada uno cuenta con sus funciones correspondientes.

En el segundo bloque se lleva a cabo la gestión de citas por parte del usuario administrador, donde se realizan diversas funciones tales como: aceptar, denegar y eliminar citas. Además de asignar un técnico especializado dependiendo del tipo de falla o servicio y él envió de un email con los detalles de la unidad y del servicio realizado. En este caso, se plantea la gestión de citas en un taller de servicio automotriz, donde el administrador tiene que realizar acciones u operaciones con respecto a cada una de las peticiones que los clientes van solicitando. Esta persona debe verificar el horario de atención, para poder aceptar una cita. Esto gracias a un botón que manda por medio de AJAX una petición al servidor con ayuda de una consulta SQL en lenguaje PHP para poder actualizar el estatus de la cita y pasar de pendiente a aceptada o cancelada, así como también asignar a un elemento o empleado del taller especialista en el servicio a realizar; ya sea servicio preventivo, correctivo o alguna falla eléctrica. Una vez que la cita se ha llevado a cabo, se cuenta con la herramienta básica de poder mandar un correo al cliente notificándole los detalles del servicio, así como también el costo total del mismo.

En el manejo de los servicios que se ofrecen en la aplicación, se pueden cometer errores al ingresarlos o que quizás con el pasar del tiempo los precios se adapten a las necesidades del mercado. Así que se cuenta con un apartado donde se puedan introducir nuevos servicios, actualizar sus precios o descripciones, corregir errores y eliminar algunos ya obsoletos; de tal manera, que el administrador no tiene que actualizar el sitio, sino que sus acciones se verán reflejadas al momento de hacer los cambios.

El tercer bloque engloba la gestión de un chat entre cliente y administrador, en el que se despejan dudas y se brinda información adicional. Sin duda una herramienta muy efectiva para brindar una atención personalizada, que además genera un acercamiento y una comunicación más estrecha con los clientes. Este chat, al estar hecho con AJAX, se vuelve muy dinámico, ya que el usuario no necesita actualizar la página para poder ver el mensaje enviado,

solamente al apretar el botón de enviar; este se manda, y se visualiza al momento.

El cuarto bloque considera la gestión del personal, en la que el administrador es el encargado de dar de alta nuevos miembros del equipo de trabajo, actualizar sus datos, o eliminarlos del sistema si ya no se encuentran trabajando. Es aquí donde es bastante perceptible que AJAX puede ser utilizadas en más herramientas de la aplicación para poder optimizar tiempos y mejorar la usabilidad. Por ejemplo:

- El manejo del personal
- La gestión de servicios ofertados a los clientes
- Envío de formularios u opiniones
- Un Prototipo de chat entre administrador y clientes

En el manejo del personal, se ofrece la propuesta de un alta de trabajadores con un formulario de sus datos principales, además de la capacidad de poder actualizar sus datos, eliminarlos, modificarlos sin tener que actualizar la aplicación para ver si los datos han sido actualizados con éxito.

Finalmente, aunque existen muchas formas de lanzar notificaciones o alertas en una web. Una de esas formas es usando la función alert de JavaScript, que muestra una caja de alerta con el mensaje que se desee. En los tiempos que corren esta es una opción poco estética. En cuanto al manejo de estos cuadros de dialogo y de confirmación de acciones, se trabajó con SweetAlert 2. Como ya es sabido, las alertas normales sirven para mostrarle información al usuario, con el plugin SweetAlert 2 se logra darle una mejor impresión al usuario. Ya que permite mostrar alertas tipo alert usables, responsivas, personalizables y accesibles. Las alertas son una especie de ventanas modales que se adaptan perfectamente a lo que se desea en este proyecto, además de brindarle cierto dinamismo.

En las siguientes secciones se explican los diferentes elementos que constituyen la aplicación Coq's Tuning Performance, para mejorar la atención al cliente mediante el procesamiento de solicitudes en segundo plano.

4.1 Página de Inicio

El diseño de un índice activo, dinámico y responsivo es fundamental en cualquier aplicación, ya que el usuario visualizará esta pestaña cada vez que necesite realizar un trámite o servicio y, por otro lado; los internautas encontrarán un sitio agradable a la vista con información relevante y específica de acuerdo a las necesidades de cada uno; de manera eficaz, y proporcionando comodidad mientras navegan.

En esta parte se incluye el uso de Carruseles de imágenes con atractivos efectos, para dar un diseño agradable y dinámico, mientras que, por otro lado, se muestra el eslogan y un pequeño apartado de ¿sabías qué?, con recomendaciones importantes, tal como se visualiza en la figura 2.

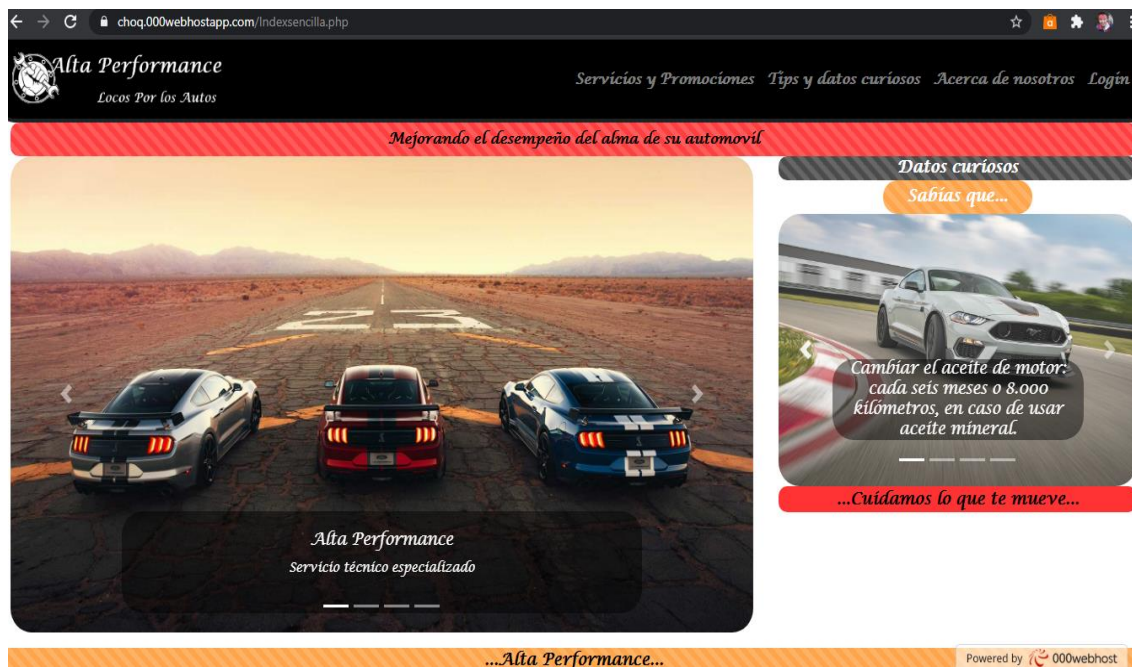


Figura 2. Índice del sistema Coq's Tuning Performance.

Esta aplicación cuenta con varias pestañas de información para los internautas y usuarios, pero la parte fundamental de este proyecto se basa en el *back-end* y eso se aprecia después de haber iniciado sesión.

4.2 Manejo de Sesiones de Usuario

Para este manejo de sesiones, se trabaja con PHP, JavaScript y SweetAlert 2 en cuanto a las alertas por envío de datos erróneos en los formularios. En la figura 3 se aprecia el formulario de inicio de sesión.

El formulario de inicio de sesión está superpuesto sobre una imagen de fondo que muestra herramientas como un martillo y destornilladores. El formulario tiene un encabezado rojo con el texto "...Iniciar sesión...". Debajo, se pide al usuario que "Ingresa tu correo y contraseña". Hay dos campos de entrada: uno etiquetado "email" y otro "Password". Debajo de estos campos hay un botón naranja con el texto "INGRESAR". A continuación, se ofrece la opción "O si no estas registrado" con un botón gris que dice "REGISTRARSE". En la parte inferior, hay un enlace que dice "Si perdiste tu contraseña has clic [aquí](#)".

Figura 3. Inicio de sesión al sistema.

El proceso para poder darse de alta en el sistema, es muy sencillo, se debe proporcionar un correo valido al cual será enviado un código de confirmación de correo con ayuda de PHP Mailer, y una vez recibido el usuario debe dar clic al botón verificar código a donde se le presentará un segundo formulario al cual debe ingresar dicho código, confirmar su correo y contraseña para poder darse de alta exitosamente.

Una vez que el registro concluye de manera exitosa, el usuario puede iniciar sesión y, en caso de olvidarla, puede recuperarla en cualquier momento.

Ya con la sesión iniciada, el usuario dispone de un menú con diversas opciones dependiendo de lo que requiera consultar o realizar. Las cuales se aprecian en la figura 4.

- Agendar una cita
- Buzón de sugerencias
- Estatus de la cita
- Chat en línea
- Cierre de sesión



Figura 4. Menú del **usuario** cliente.

4.3 Gestión de Citas

En lo que respecta al usuario administrador, donde se trabaja la gestión con ayuda de AJAX para poder aceptar, denegar, eliminar citas, asignar a algún técnico especializado de acuerdo a la falla en cuestión, y finalmente, una vez que el servicio se ha realizado de manera exitosa enviar un reporte detallado a la dirección de correo del cliente con los detalles más relevantes de la unidad y del servicio.

La interfaz con la que el administrador realizará las actividades antes mencionadas es la que se aprecia en la figura 5.

Alta Performance

Locos Por los Autos

[Servicios y Promociones](#)
[Tips y datos curiosos](#)
[Acerca de nosotros](#)
[Login](#)

...Citas del mes...

Asignar mecánico

Nombre del mecánico

Antonio Pérez Vázquez

ASIGNAR MECÁNICO

Asignar costo y Descripción

Costo del Servicio

Añada una descripción del Servicio

ENVIAR MAIL

Cliente	Marca	Vehículo	Modelo	Servicio	Fecha	Hora	Estatus	Mecánico	Eliminar	Aceptar	Denegar
franaxla7x@gmail.com	Ford	Mustang	2020	Afinacion Menor	2020-06-13	02:00	Aceptada	Antonio Pérez Vázquez	ELIMINAR	ACEPTAR	CANCELAR
chava.ft@hotmail.com	Audi	TT	2020	Afinacion mayor	2020-06-18	11:00	Aceptada	Frank Ordaz Robles	ELIMINAR	ACEPTAR	CANCELAR
isabelsot@outlook.com	Audi	A3	1999	Afinacion Menor	2020-06-08	12:00	Aceptada	Frank Ordaz Robles	ELIMINAR	ACEPTAR	CANCELAR
antonio.pe.v@hotmail.com	Volkswagen	Bettle	2017	Falla	2020-06-05	01:00	Aceptada	Antonio Pérez Vázquez	ELIMINAR	ACEPTAR	CANCELAR

...Alta Performance...

000webhost

Figura 5. Manejo de citas por parte del Administrador.

Para lograr esta funcionalidad se trabajó en documentos diferentes. Para hacer el diseño y maquetado del *template* o plantilla de las citas del mes, se utilizó código HTML, PHP, y CSS con Bootstrap.

El segundo fue un documento JavaScript, donde se crearon las funciones para cada botón y para cada formulario de llenado.

Iniciando con la función mostrar tabla, en la cual se lista la tabla de la base de datos donde se encuentran almacenadas las citas (vea la figura 6).

```

.php      x
app.js    x

function mostrartabla(){
    $.ajax({
        url: 'listatarea.php',
        type: 'GET',
        success: function(response){
            let tareas = JSON.parse(response);
            let template = '';

```

Figura 6. Función mostrar tabla.

La cual envía la petición a un tercer documento llamado listatarea.php a donde se encuentran las consultas (*Querys*) para mandar a imprimir los datos, tal como se visualiza en la figura 7.

```
include 'conexionagenda.php';
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
$query = "SELECT * from agenda2";
$result = mysqli_query($connection,$query);
if (!$result) {
    die('fallo en la conexion'.mysqli_error($connection));
}
$json = array();
while($row= mysqli_fetch_array($result)){
    $json[] = array(
        'email' => $row['email'],
        'marca' => $row['marca'],
        'modelo' => $row['modelo'],
        'anio' => $row['anio'],
        'servicio' => $row['servicio'],
        'fecha' => $row['fecha'],
        'hora' => $row['hora'],
        'estatus' => $row['estatus'],
        'mecanico' => $row['mecanico'],
        'idp' => $row['idp']
    );
}
$jsonstring =json_encode($json);
echo "$jsonstring";
```

Figura 7. Documento que lista las citas.

Una vez que la consulta es exitosa, el documento listatarea.php, regresa en un array los datos de la agenda y los manda a la interfaz correspondiente. Nuevamente, se utiliza el documento app.js que envía a plotear los datos del array en la tabla dentro de la función mostrartabla() en la figura 8.

```
success: function(response){
    let tareas = JSON.parse(response);
    let template = '';
    tareas.forEach(tareas => {
        template +=
        `<tr tareaid="${tareas.idp}" class="table-default">
            <th scope="row">
                <a href="#" class="tarea-item" style="color:black;">${tareas.email}
            </th>
            <td>${tareas.marca}</td>
            <td>${tareas.modelo}</td>
            <td>${tareas.anio}</td>
            <td>${tareas.servicio}</td>
            <td>${tareas.fecha}</td>
            <td>${tareas.hora}</td>
            <td>${tareas.estatus}</td>
            <td>${tareas.mecanico}</td>
```

Figura 8. Template de citas.

La respuesta del array se asigna a la variable tareas, y se accede a cada una de sus posiciones para mandarlo a la tabla en pantalla. También se declaran los botones que ayudan a gestionar la cita.

4.3.1 Eliminación de una cita

Se utiliza una función de tipo onclick, la cual está a la espera de que ocurra el suceso de apretar el botón eliminar, y una vez que esto ocurre se obtiene el id del registro para poder realizar la eliminación del mismo, no sin antes lanzar una alerta del tipo SweetAlert al administrador, donde podrá confirmar o cancelar la acción. Esto con el fin de evitar un accidente y que se borre un registro por error, tal como se aprecia en la figura 9.

```
$(document).on('click','eliminar',function () {
    let elemento = $(this)[0].parentElement.parentElement;
    let id = $(elemento).attr('tareaid');
    Swal.fire({
        title: '¿Estás seguro?',
        text: "¡No podrás revertir esto!",
        icon: 'warning',
        showCancelButton: true,
        confirmButtonColor: '#3085d6',
        cancelButtonColor: '#d33',
        confirmButtonText: 'Si, aceptar'
    }).then((result) => {
        //
        if (result.isConfirmed) {
            Swal.fire(
                'Eliminado!',
                'Esta cita ha sido eliminada',
                'success'
            )
            //Aquí vamos a enviar a eliminar, el id de la
            $.post('eliminar.php',{id},function (response) {
                console.log(response);
                mostrartabla();
            })
        }
    })
} else if (
    result.dismiss === Swal.DismissReason.cancel
) {
    Swal.fire(
        'Interrumpido',
        'La cita está a salvo :)',
        'warning'
    )
}
})
})
```

Figura 9. Función eliminar.

En caso de confirmar esta acción, se envía el id del registro al documento eliminarar.php, en el cual, una vez que establece la conexión con la base de datos en el servidor, realiza la consulta (*Query*) de eliminación de dicho registro y devuelve en consola si fue exitoso o no, y una alerta SweetAlert para que sea más visual. En la figura 10 se aprecia la consulta SQL.

```
include 'conexionagenda.php';
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
if (isset($_POST['id'])) {
    $id=$_POST['id'];
    $query = "DELETE from agenda2 WHERE idp=$id";
    $result = mysqli_query($connection,$query);
    if(!$result){
        die('Error en la consulta de eliminacion....'. mysqli_error($connection));
    }
    echo "Tarea eliminada con facilidad";
}
```

Figura 10. Consulta SQL eliminar.

4.3.2 Aceptación de una cita

Se utiliza del mismo modo una función de tipo onclick, la cual está a la espera de que ocurra el suceso de apretar el botón aceptar, y una vez que esto ocurre se obtiene el id del registro para poder realizar la actualización del mismo, tal como se aprecia en la figura 11.

```
//Función aceptar inicia
$(document).on('click','.aceptar',function () {
    let elemento = $(this)[0].parentElement.parentElement;
    console.log(elemento);
    let id = $(elemento).attr('tareaid');
    console.log(id);
    Swal.fire({
```

Figura 11. Función aceptar.

Todo ello, no sin antes lanzar al administrador una alerta del tipo SweetAlert donde podrá confirmar o cancelar la acción. Esto con el fin de evitar

un accidente y que se acepte una cita, aun cuando no es posible por exceso de saturación. En la figura 12 se aprecia el funcionamiento y estructura de la alerta.

```
Swal.fire({
  title: '¿Deseas aceptar esta cita?',
  text: ":"),
  icon: 'warning',
  showCancelButton: true,
  confirmButtonColor: '#3085d6',
  cancelButtonColor: '#d33',
  confirmButtonText: 'Si, aceptar'
}).then((result) => {
  //
    if (result.isConfirmed) {
      Swal.fire(
        'Aceptada!',
        'Esta cita ha sido aceptada',
        'success'
      )
      //Aquí vamos a enviar a editartar, el id de la fila
      $.post('editartar.php',{id},function (response) {
        console.log(response);
        mostrartabla();
      })
    }else if (
      result.dismiss === Swal.DismissReason.cancel
    ) {
      Swal.fire(
        'Atención',
        'La cita sigue pendiente :)',
        'warning'
      )
    }
  })
})
```

Figura 12. Estructura de las alertas de confirmación.

Una vez que el usuario acepta la cita, se manda el id de la misma al documento editartar.php a donde se realizará una consulta (*Query update*) para actualizar el estatus de pendiente a aceptada y de este modo, tanto el administrador como el cliente podrán visualizarla, tal como se ve en la figura 13.

```

include 'conexionagenda.php';
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
if (isset($_POST['id'])) {
    $id=$_POST['id'];
    $query = "UPDATE agenda2 SET estatus='Aceptada' WHERE idp=$id";
    $result = mysqli_query($connection,$query);
    if(!$result){
        die('Error en la consulta de edición....'. mysqli_error($connection));
    }
    echo "Tarea editada con facilidad";
}

```

Figura 13. Consulta SQL actualizar.

4.3.3 Alerta Push.

Cabe mencionar que se tiene una función alerta de tipo push, la cual se ejecuta cada que se actualiza el sitio y verifica con AJAX, si existe algún registro en la base de datos que esté en estatus de pendiente, mostrará una alerta push que le avisará al administrador que tiene una cita pendiente que debe atender. Esta función ayuda muchísimo, ya que, si se cuentan con muchas citas en pantalla, es difícil percibir su estatus. En cambio, con esta alerta se visualizará y también generará un sonido peculiar al aparecer.

```

72     function alerta(){
73         console.log('alerta');
74         $.ajax({
75             url: 'alerta.php',
76             type: 'GET',
77             success: function(response){
78                 console.log(response);
79                 if (response=='pendiente') {
80                     Push.create("Pedido Nuevo",{
81                         body: "Tienes Pedidos sin atender",
82                         icon: "Notifications/alerta.png",
83                         timeout: 10000,
84                     });
85                 }
86             }
87         });
88     });
89 }

```

Figura 14. Función alerta.

En la figura 15 se apreciar una alerta de tipo banner. La cual aparece a un costado del escritorio en el ordenador generando el típico sonido de notificación, con un mensaje configurable y con una imagen de alerta predeterminada por el usuario.

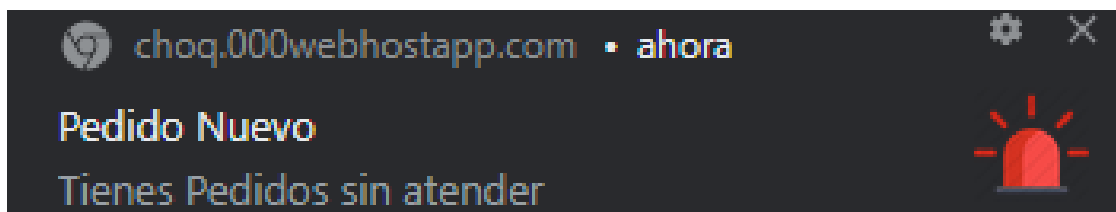


Figura 15. Visualización de la notificación tipo Push.

Si el usuario administrador le da un clic a esta notificación, lo llevará directamente al sitio web específicamente a la página de gestión de citas donde podrá encontrar fácilmente la cita que debe ser atendida.

4.3.4 Cancelación de una cita

La función de cancelar una cita es muy similar a la de aceptar, ya que es una función onclick que se encuentra a la espera de que suceda el evento para poder obtener el id de dicha cita y antes de cancelarla, pregunta al administrador por medio de un SweetAlert si realmente desea cancelarla. En la figura 16 se aprecia dicha función.

De ser así, manda el id obtenido al documento cancelar.php, en el cual genera una Query update que actualiza el estatus a cancelado, avisando al usuario cliente que existe un conflicto en ese horario e invitándolo a agendar en otro horario. Tal como se visualiza en la figura 17. En caso contrario, se manda una SweetAlert donde se notifica que la cita sigue pendiente a la espera de ser aceptada.

```

$(document).on('click', '.cancelar', function () {
    let elemento = $(this)[0].parentElement.parentElement;
    console.log(elemento);
    let id = $(elemento).attr('tareaid');
    console.log(id);
    Swal.fire({
        title: '¿Deseas cancelar esta cita?',
        text: ":)",
        icon: 'warning',
        showCancelButton: true,
        confirmButtonColor: '#3085d6',
        cancelButtonColor: '#d33',
        confirmButtonText: 'Si, cancelar'
    }).then((result) => {
        if (result.isConfirmed) {
            Swal.fire(
                'Cancelada!',
                'Esta cita ha sido Cancelada por saturación',
                'success'
            )
            //Aquí vamos a enviar a cancelar, el id de la fila del
            $.post('cancelar.php',{id},function (response) {
                console.log(response);
                mostrartabla();
            })
        }
        else if (
            result.dismiss === Swal.DismissReason.cancel
        ) {
            Swal.fire(
                'Atención',
                'La cita sigue pendiente :)',
                'warning'
            )
        }
    })
})

```

Figura 16. Función cancelar.

```

include 'conexionagenda.php';
$conection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
if (isset($_POST['id'])) {
    $id=$_POST['id'];
    $query = "UPDATE agenda2 SET estatus='Cancelada/ estamos saturados a esa hora'";
    $result = mysqli_query($conection,$query);
    if(!$result){
        die('Error en la consulta de edición....'. mysqli_error($conection));
    }
    echo "Tarea editada con facilidad";
}

```

Figura 17. Query de cancelación.

4.3.5 Asignación de un técnico

En el caso de asignar un técnico de acuerdo a la falla o servicio a realizar, se logra gracias a otra función de tipo onclick, en la cual, por medio de la clase, se puede obtener todos sus datos. Así como también mandarlos a pantalla y llenar un formulario. Tal es el caso de la asignación de un técnico especializado de acuerdo a la falla o servicio (vea la figura 18).



Figura 18. Asignación de empleado.

El administrador debe dar clic al correo del cliente que aparece en pantalla y en automático, el formulario de la parte posterior se llena con los datos de dicha cita. La cual por default tiene el valor por asignar y el administrador puede seleccionar directamente de la base de datos a cualquiera de los empleados de acuerdo a su especialidad o carga laboral. Visualizando la estructura de la función, así obtiene el id de la cita y lo envía al documento asignam-simple.php el cual se encarga de obtener los datos de ese registro en la base y los devuelve en pantalla los datos. Vea la figura 19 de la función de asignación.

```

212     $(document).on('click', '.tarea-item', function () {
213         let elemento = $(this)[0].parentElement.parentElement;
214         console.log(elemento);
215         let id = $(elemento).attr('tareaid');
216         console.log(id);
217         $.post('asignam-simple.php', {id}, function(response){
218             console.log(response);
219             const tarea = JSON.parse(response);
220             console.log(tarea);
221             console.log(tarea.mecanico);
222             $('#mecanico').val(tarea.mecanico);
223             $('#idtar1').val(tarea.idp);
224             edit = true;
225         })
226     });

```

Figura 19. Función asignar mecánico.

Después de hacer la selección se le da clic al botón asignar y en ese momento entra la siguiente función: figura 20.

```

8     $('#task-form').submit(function(e) {
9         const postData = {
10             mecanico: $('#mecanicoa').val(),
11             id: $('#idtar1').val()
12         };
13         console.log("AAAAAAAAAAAA");
14         console.log(postData);
15         let url = edit === false ? 'agregarmec.php' : 'mecanico-asig.php';
16         console.log(url);
17         $.post(url, postData, function(response){
18             console.log(response);
19             mostrartabla();
20             $('#task-form').trigger('reset');
21         });
22         e.preventDefault();
23     });

```

Figura 20. Lectura de la variable edit.

Esta función cuenta con el id del formulario de asignación de mecánico, en cual se pueden hacer 2 funciones, de las cuales solo se habilitará una, que es la de asignar un mecánico a la cita o servicio. La segunda función sirve para crear registros nuevos en la base de datos, pero en este caso eso se hará en otro apartado donde se agregarán nuevos empleados y donde se pueden editar los ya existentes.

Esto se realiza leyendo la variable edit. Si es verdadera, entonces realizara la asignación de un técnico. El valor verdadero es asignado en la función asignar mecánico

Así que, en este caso, solo se envía el id y el nombre del mecánico al documento mecanico-asign.php en el que se ejecuta una consulta de actualización e inserta el nombre en el registro de la cita. Tal como se aprecia en la figura 21.

```
include 'conexionagenda.php';
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
$id = $_POST['id'];
$mecanico = $_POST['mecanico'];

$query = "UPDATE agenda2 set mecanico='$mecanico' WHERE idp='$id'";
$result= mysqli_query($connection,$query);
if(!$result){
    die('Error en la consulta de edicion....'. mysqli_error($connection));
}
echo "Todo al cien con la edicion";
echo $id;
echo $mecanico;
```

Figura 21. Query que asigna un mecánico

4.3.6 Envío de correos

Esta función es muy similar a la anterior de asignar un técnico, ya que se debe dar clic al correo de la cita para que por medio de una función se obtenga el id y buscarlo en la base de datos para devolver los valores de dicha cita

En la figura 22 se muestra el pequeño formulario, en el cual, el administrador deberá dar clic primero al email del correo del cliente. Y después ingresar de modo manual el costo total de la reparación y una breve descripción del servicio realizado.

Figura 22. Formulario de detalles del servicio

Este formulario cuenta también con unos inputs de tipo *hidden*, los cuales tienen la función de verificar que realmente al darle clic a alguna cita en la agenda, se devuelvan los datos de dicho servicio. Esto es con el fin de poder incluirlos en el correo electrónico sin tener que mostrar todo el formulario lleno de dichos campos. En la figura 23 se aprecian dichos campos hidden con los datos de la cita

```
<form id="mail-form" action="insertar_mail.php" method="POST">
  <input type="hidden" id="idtar" name="idtar">
  <input type="hidden" id="marca" name="marca">
  <input type="hidden" id="modelo" name="modelo">
  <input type="hidden" id="anio" name="anio">
  <input type="hidden" id="servicio" name="servicio">
  <input type="hidden" id="mecanico2" name="mecanico2">
  <div class="form-group">
    <input type="hidden" id="cliente" name="cliente" placeholder="correo"
    form-control">
    <label id="lab" style="color: black; font-size: medium; position: absolute; top: 255px; left: 255px; width: 255px; height: 255px; border: 1px solid black; border-radius: 50%; display: flex; align-items: center; justify-content: center; text-align: center; font-weight: bold; font-size: 1.2em; color: red; background-color: white; z-index: 1000;"> Asignar costo y Descripción </label>
    <input type="number" id="costo" name="costo" placeholder="Costo del Servicio" />
  </div>
  <div class="form-group">
    <textarea id="descripcion" name="descripcion" cols="20" rows="2" class="form-control" placeholder="Añada una descripción del Servicio"></textarea>
  </div>
  <button type="submit" class="btn btn-danger btn-sm btn-block" style="background-color: red; color: white; padding: 5px 10px; border: none; border-radius: 5px;">
    >Enviar mail</button>
</form>
```

Figura 23. Campos Hidden del formulario

Donde se requiere nuevamente una función similar de tipo onclick que obtiene los datos del servicio enviando el capturado al documento asignaem.php (ver figura 24).

```
$(document).on('click','tarea-item',function () {  
    let elemento = $(this)[0].parentElement.parentElement;  
    console.log(elemento);  
    let id = $(elemento).attr('tareaid');  
    console.log(id);  
});
```

Figura 24. Función Datos de la cita

Esto se hace para que devuelva el array de datos de la cita por medio de una variable de tipo JSON, como se visualiza en la figura 25, donde se recaban los datos del vehículo que más adelante se enviarán vía email.

```
include 'conexionagenda.php';  
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);  
$id=$_POST['id'];  
$query = "SELECT * FROM agenda2 where idp=$id";  
$result = mysqli_query($connection,$query);  
if (!$result) {  
    die('fallo en la edicion de la baseeee'.mysqli_error($connection));  
}  
$json= array();  
while ($row = mysqli_fetch_array($result)) {  
    $json[] = array(  
        'email' => $row['email'],  
        'marca' => $row['marca'],  
        'modelo' => $row['modelo'],  
        'anio' => $row['anio'],  
        'servicio' => $row['servicio'],  
        'fecha' => $row['fecha'],  
        'hora' => $row['hora'],  
        'estatus' => $row['estatus'],  
        'mecanico' => $row['mecanico'],  
        'idp' => $row['idp']  
    );  
}  
$jsonstring = json_encode($json[0]);  
echo $jsonstring;  
?>
```

Figura 25. Extracción de los datos de la cita

Después de haber obtenido el conjunto de datos, se mandan a imprimir en consola para verificar que los resultados obtenidos sean correctos, tal como se aprecia en la figura 26. Y después son asignados a las secciones del formulario que están ocultos para no saturar la vista del formulario del correo.

```
$.post('asignaem.php',{id},function(response){
    console.log(response);
    const tarea = JSON.parse(response);
    console.log(tarea);
    console.log(tarea.idp);
    console.log(tarea.email);
    console.log(tarea.marca);
    console.log(tarea.modelo);
    console.log(tarea.anio);
    console.log(tarea.servicio);
    console.log(tarea.mecanico);
    $('#cliente').val(tarea.email);
    $('#idtar').val(tarea.idp);
    $('#marca').val(tarea.marca);
    $('#modelo').val(tarea.modelo);
    $('#anio').val(tarea.anio);
    $('#servicio').val(tarea.servicio);
    $('#mecanico2').val(tarea.mecanico);
    edit = true;
});
```

Figura 26. Impresión en consola de los datos de la BD

Finalmente, al dar clic al botón del formulario se enviará un correo electrónico con los detalles de la cita, como son: Marca, modelo, año, servicio, mecánico a cargo del servicio, costo y descripción del servicio realizado. Todo esto con un maquetado con HTML para darle un buen formato tal como se ve en la figura 27.

Así es como funciona este procesamiento de solicitudes en segundo plano, ya que el usuario no percibe todas las consultas SQL y peticiones al servidor que se realizan con tan solo dar clic en un botón o en un link y por otro lado no es necesario actualizar o volver a cargar la página para visualizar los cambios realizados con las solicitudes de citas o con la gestión de las mismas. Con este conjunto de herramientas se trabajaron otras partes fundamentales de esta aplicación, para poder ofrecer una mayor versatilidad y comodidad al momento de realizar estas tareas, ya que hoy en día, todas las aplicaciones de software

trabajan casi en su totalidad en tiempo real y se actualizan los datos constantemente.

```
56 $message .= "Vehículo: " . $marca;  
57 $message .= '<br>';  
58 $message .= "Modelo: " . $modelo;  
59 $message .= '<br>';  
60 $message .= "Año: " . $anio;  
61 $message .= '<br>';  
62 $message .= "Servicio realizado: " . $servicio;  
63 $message .= '<br>';  
64 $message .= "mecánico a cargo de la unidad: " . $mecanico;  
65 $message .= '<br>';  
66 $message .= "Costo de servicio: $" . $costo;  
67 $message .= '<br>';  
68 $message .= "Detalles: " . $descripcion;  
69  
70 $message .= '</p>';  
71 $message .= '</body><p>Gracias por tu confianza</p></html>';  
72 //$message->charset = 'utf8';  
73 $mail = mail($mail para, $subject, utf8_decode($message), $header);
```

Figura 27. Estructura del mensaje

El lenguaje PHP es muy poderoso y sencillo de implementar, como programador puedo decir que; después de haber trabajado con JavaServlets, me gusta más PHP. Y aunque ambos tienen sus ventajas, PHP es más amigable y fácil de implementar.

Por otro lado, si a PHP se une con Java Script se pueden crear funciones y herramientas más poderosas. Tal es el caso del siguiente chat creado con Ajax. En el cual se aprecian sus aplicaciones.

4.4 Chat Con Ajax y PHP

Como todo negocio, lo ideal es ganar más dinero. Pero para lograr ese objetivo, es necesario tener clientes, ir ganando nuevos clientes y por supuesto, conservar a los que se tienen. Una manera de conseguirlo es conociendo bien a dichos clientes. Y la mejor forma, además de formularios de retroalimentación en

cuanto a su opinión de un servicio, es un espacio virtual donde se puedan resolver sus dudas, opiniones o problemas que puedan surgir.

Es por eso que después de haber estudiado las técnicas AJAX, se desarrolló el siguiente chat que se aprecia en la figura 28, para crear un enlace o canal en el que puedan comunicarse el administrador y el cliente con el fin de resolver dudas o aclarar alguna situación.

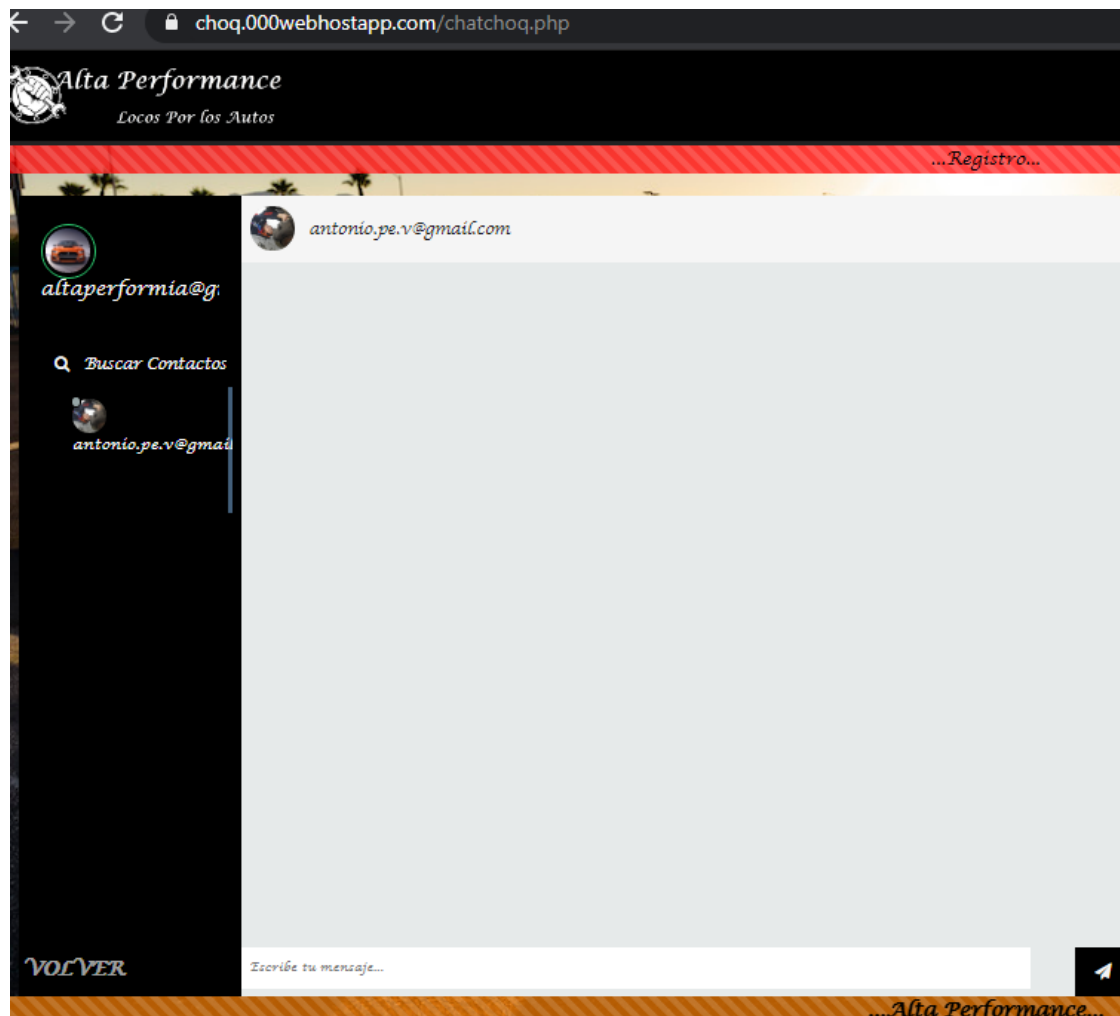


Figura 28. Vista del chat entre usuario y administrador

4.4.1 Maquetado del chat

Lo primero que se desarrolla es el maquetado del chat, donde se incluye la caja de mensajes, el perfil del usuario con una imagen que se asigna por default al registrarse en el sistema y, en el caso del administrador: Una sección

lateral donde se muestran los clientes para poder ir monitoreando las conversaciones y brindar una respuesta rápida. Tal como se aprecia en la figura 29 que es la vista por parte del administrador.

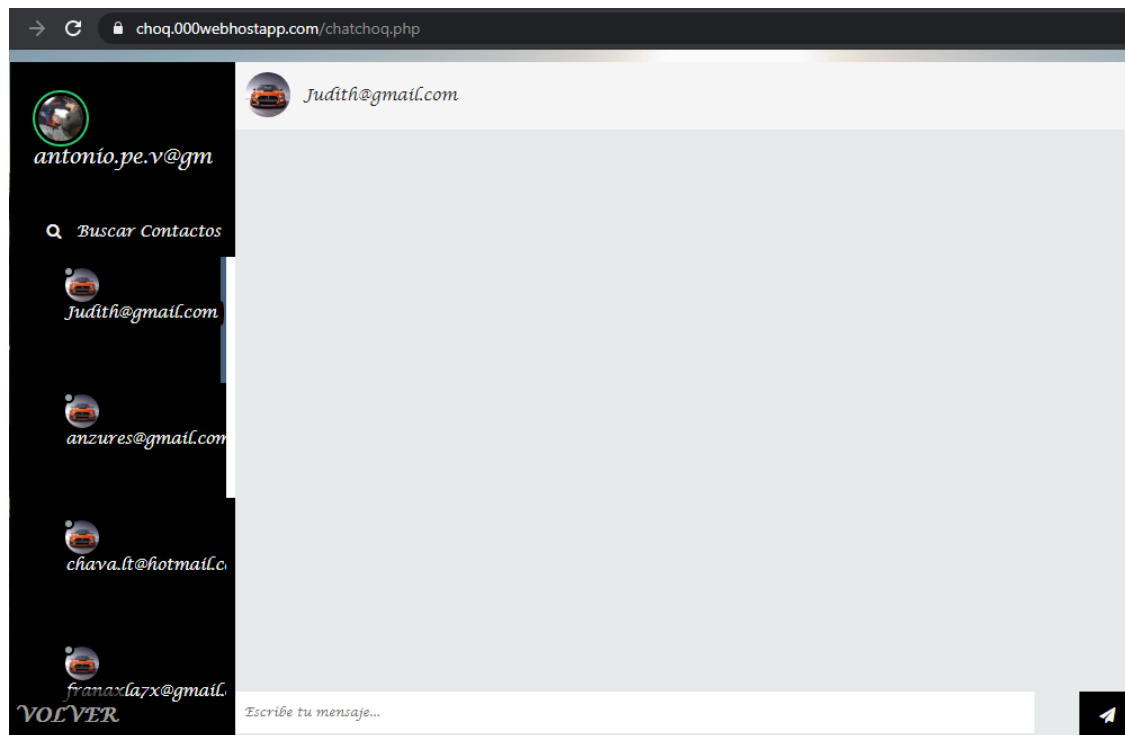


Figura 29. Vista del chat entre administrador y usuarios

Se utiliza una función que despliega la lista de usuarios para nuestro chat, en la cual. Se analiza el id de usuario que tiene la sesión iniciada para determinar si es usuario o administrador.

Se hace uso de AJAX y se envía al documento chat_action.php un dato de tipo JSON, una cadena update_user_list. Y como respuesta, devuelve la base de datos, los usuarios con su imagen de perfil, y un pequeño botón indicador, el cual si está en verde indica que el usuario está en línea y si está en gris se entiende que está desconectado; tal como se muestra en la figura 30 el respectivo código de la función.

```

function updateUserList() {
    console.log("entrando al update user list");
    $.ajax({
        url: "chat_action.php",
        method: "POST",
        dataType: "json",
        data: {action: 'update_user_list'},
        success: function(response) {
            console.log("respuesta" + response);
            var obj = response.profileHTML;
            console.log("obj" + obj);
            Object.keys(obj).forEach(function(key) {
                // update user online/offline status
                if($("#" + obj[key].userid).length) {
                    if(obj[key].online == 1 && !$("#status_" + obj[key].userid).hasClass('online'))
                        $("#status_" + obj[key].userid).addClass('online');
                    else if(obj[key].online == 0) {
                        $("#status_" + obj[key].userid).removeClass('online');
                    }
                }
            });
        }
    });
}

```

Figura 30. Función updateUserList.

En la figura 31 se aprecia como aparece en pantalla un usuario conectado y uno ausente. Esto es muy útil, ya que así, no solamente se tiene un canal de comunicación entre el cliente y el administrador, sino que también es posible saber si se encuentra disponible.

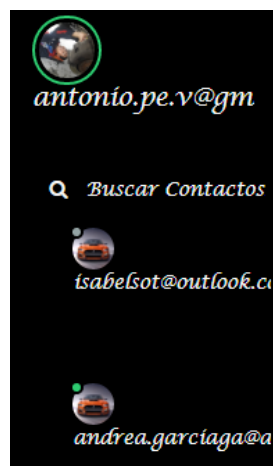


Figura 31. Usuario en línea y desconectado.

4.4.2 Estatus Typing

Ahora se crea otra función, muy útil, conocida comúnmente como escribiendo (typing), la cual trabaja tal como lo dice su nombre, cuando un

usuario da clic en el placeholder para escribir su mensaje, se manda a la base de datos en la tabla detalles una actualización de no a yes.

Después se realiza una condición para obtener el estatus del usuario, y si es positivo, se manda a pantalla una pequeña leyenda que indica que está escribiendo, tal como se muestra en la figura 32.

```
180 public function updateTypingStatus($is_type, $loginDetailsId) {
181     $sqlUpdate = "
182         UPDATE ".$this->chatLoginDetailsTable."
183         SET is_typing = '".$is_type."'
184         WHERE id = '".$loginDetailsId."'";
185     mysqli_query($this->dbConnect, $sqlUpdate);
186 }
187 public function fetchIsTypeStatus($userId){
188     $sqlQuery = "
189     SELECT is_typing FROM ".$this->chatLoginDetailsTable."
190     WHERE userid = '".$userId."' ORDER BY last_activity DESC LIMIT 1";
191     $result = $this->getData($sqlQuery);
192     $output = '';
193     foreach($result as $row) {
194         if($row["is_typing"] == 'yes'){
195             $output = ' - <small><em>escribiendo...</em></small>';
196         }
197     }
198     return $output;
199 }
```

Figura 32. Función de estatus de escritura.

Aquí se aprecia en la figura 33 la palabra escribiendo para informar al usuario receptor que el emisor está escribiendo un mensaje.



Figura 33. Visualización del estatus de escritura

4.4.3 Manejo de Mensajes

La parte medular del chat, que es la inserción de mensajes y el despliegue de los mismos de acuerdo al tiempo de envío. Para ello se trabaja de

la siguiente manera con ayuda de AJAX, PHP, MySQL y HTML para el maquetado de los contenedores de cada mensaje.

Iniciando con la función que va a mostrar el chat de usuario, la cual manda al documento chat_action.php por método post el id del usuario receptor y la cadena show chat tal como se muestra en la figura 34.

```
115 function showUserChat(to_user_id){
116     $.ajax({
117         url:"chat_action.php",
118         method:"POST",
119         data:{to_user_id:to_user_id, action:'show_chat'},
120         dataType: "json",
121         success:function(response){
122             $('#userSection').html(response.userSection);
123             $('#conversation').html(response.conversation);
124             $('#unread_'+to_user_id).html('');
125         }
126     });
127 }
```

Figura 34. Función showUserChat.

Se utiliza una condicional if para detectar la cadena que se está recibiendo y al ser show_chat manda al documento chat.php el id de la sesión del usuario y el id del usuario al que va a ser enviado el mensaje tal como se muestra en la figura 35.

```
15 if($_POST['action'] == 'show_chat') {
16     $chat->showUserChat($_SESSION['id'], $_POST['to_user_id']);
17 }
```

Figura 35. Condicional show_chat.

Después entra a la función ShowUserChat la cual se encarga de mostrar el chat en pantalla, obtiene la foto de perfil del usuario y el correo electrónico para mostrarlo en la sección correspondiente en la cabecera del chat.

Esta es la respuesta que recibe la función de AJAX y es lo que manda a imprimir en nuestra sección del chat

Además, actualiza la tabla de la base de datos donde se almacena el chat entre el administrador y el usuario o viceversa, cambia el estatus a 0 para

indicar que el mensaje se ha visto, ya que cuando se envía un mensaje y se inserta en la tabla el estatus se inicializa con 1. Y después actualiza la tabla de los usuarios para saber si se ha iniciado un chat o no y para obtener el id del ultimo usuario con el que se estuvo chateando tal como se aprecia en la figura 36.

```
public function showUserChat($from_user_id, $to_user_id) {
    $userDetails = $this->getUserDetails($to_user_id);
    $toUserAvatar = '';
    foreach ($userDetails as $user) {
        $toUserAvatar = $user['avatar'];
        $userSection = '
        <p>' . $user['email'] . '</p>'
        ;
    }
    // get user conversation
    $conversation = $this->getUserChat($from_user_id, $to_user_id);
    // update chat user read status
    $sqlUpdate = "
        UPDATE ".$this->chatTable."
        SET status = '0'
        WHERE sender_userid = '". $to_user_id ."' AND reciever_userid = '". $f
        = '1'";
    mysqli_query($this->dbConnect, $sqlUpdate);
    // update users current chat session
    $sqlUserUpdate = "
        UPDATE ".$this->chatUsersTable."
        SET current_session = '". $to_user_id ."'
        WHERE id = '". $from_user_id ."'";
    mysqli_query($this->dbConnect, $sqlUserUpdate);
    $data = array(
        "userSection" => $userSection,
        "conversation" => $conversation
    );
    echo json_encode($data);
}
```

Figura 36. Segunda Función showUserChat

La siguiente función es getUserChat, la cual se encarga de mandar a traer de la tabla de la base de datos chat todos los registros almacenados que coincidan con el usuario que envía y que recibe o viceversa, dependiendo de quién es el emisor y receptor en cada sesión. El usuario que envía coincide con su Id propio y el que recibe con el Id del usuario al que se envía el mensaje.

Después dentro de la sección de chats se realiza un foreach() para recorrer los registros de la tabla que cumplan con los datos devueltos de la consulta SQL y de acuerdo a su tipo de usuario (emisor o receptor) se manda a

llamar la foto de perfil y el mensaje para después acomodarlos hacia la izquierda o derecha.

Aquí se investigó con respecto a una herramienta que ayude a ordenar de acuerdo al tiempo o fecha de registro de cada mensaje, la función ORDER BY timestamp asc para así poder mostrarlos en orden ascendente conforme fueron almacenados en la base de datos. tal como se aprecia en la figura 37.

```
public function getUserChat($from_user_id, $to_user_id) {
    $fromUserAvatar = $this->getUserAvatar($from_user_id);
    $toUserAvatar = $this->getUserAvatar($to_user_id);
    $sqlQuery = "
        SELECT * FROM ".$this->chatTable."
        WHERE (sender_userid = '".$from_user_id.'"
        AND reciever_userid = '".$to_user_id."')
        OR (sender_userid = '".$to_user_id.'"
        AND reciever_userid = '".$from_user_id."')
        ORDER BY timestamp ASC";
    $userChat = $this->getData($sqlQuery);
    $conversation = '<ul>';
    foreach($userChat as $chat){
        $user_name = '';
        if($chat["sender_userid"] == $from_user_id) {
            $conversation .= '<li class="sent">';
            $conversation .= '';
            $conversation .= '';
        $conversation .= '</li>';
    }
    $conversation .= '</ul>';
    return $conversation;
}
```

Figura 37. Función getUserChat.

4.4.4 Mensajes sin leer

Finalmente, se desarrolló una última función que ayuda mucho a los usuarios a saber si les ha llegado un mensaje nuevo en el chat y puedan visualizarlo a la brevedad. Esta función trabaja al igual que las anteriores con AJAX, PHP, MySQL y HTML con CSS para darle un estilo al contador de mensaje.

Lo primero que se lleva a cabo es la función en AJAX que contará los mensajes y los mostrará en pantalla `updateUnreadMessageCount()`, Figura 38. Esta función manda al documento `chat_action.php` por método POST el id del usuario al que se le envía el mensaje y la cadena `update_unread_message`. Figura 39. Aquí se encontrará una condicional en la que entrará, y una vez ahí obtiene el ID del receptor del mensaje (previamente enviado por la función AJAX) y el ID del emisor este último por medio de la función PHP `$_SESSION['id']` y manda estos datos al documento `chat.php`.

```
function updateUnreadMessageCount() {
    $('li.contact').each(function(){
        if(!$(this).hasClass('active')) {
            var to_user_id = $(this).attr('data-touserid');
            $.ajax({
                url:"chat_action.php",
                method:"POST",
                data:{to_user_id:to_user_id, action:'update_unread_message'},
                dataType: "json",
                success:function(response){
                    console.log("Entrando a los mensajes sin leer");
                    if(response.count) {
                        console.log("entrando al if contador");
                        $('#unread_'+to_user_id).html(response.count);
                        console.log("response.count"+response.count);
                    }
                }
            });
        }
    });
}
```

Figura 38. Función mensaje sin leer.

```
if($_POST['action'] == 'update_unread_message') {
    $count = $chat->getUnreadMessageCount($_POST['to_user_id'], $_SESSION['id']);
    $data = array(
        "count" => $count
    );
    echo json_encode($data);
}
```

Figura 39. Condicional `updateUnreadMessageCount`

En `chat.php` entra a la función `getUnreadMessageCount`, donde se realiza una consulta SQL para encontrar los mensajes sin leer con estatus igual

a 1 que coincidan con los IDs del emisor y el receptor. Y con ayuda de la función `getNumRows`, se obtiene el número de mensajes sin leer; y si es mayor a cero, devuelve la salida con ese valor. Tal como se aprecia en la figura 40

```
public function getUnreadMessageCount($senderUserid, $recieverUserid) {
    $sqlQuery = "
        SELECT * FROM ".$this->chatTable."
        WHERE sender_userid = '$senderUserid'
        AND reciever_userid = '$recieverUserid' AND status = '1'";
    $numRows = $this->getNumRows($sqlQuery);
    $output = '';
    if($numRows > 0){
        $output = $numRows;
    }
    return $output;
}
```

Figura 40. Función `getUnreadMessageCount`

Una vez devuelto este valor, la función AJAX recibe el número de mensajes sin leer y los manda a imprimir en pantalla a un costado de la foto de perfil del usuario con un color blanco de fuente y rojo de fondo para que se aprecie mejor. Y esta es la visualización de un chat que acaba de recibir un mensaje nuevo en la figura 41.

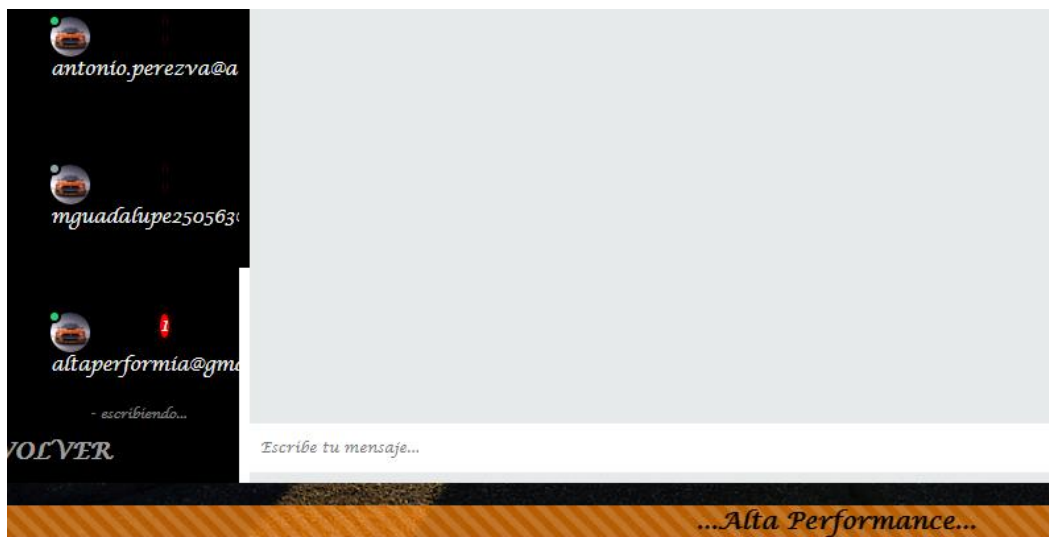


Figura 41. Vista general de un chat con mensajes sin leer

Este fue el resultado final del chat. Un trabajo que involucró varios problemas o situaciones a resolver para conseguir las funcionalidades antes mencionadas. Estos problemas se fueron resolviendo paso a paso y de manera

independiente Ya que primero se investigó respecto al funcionamiento de AJAX y ver la manera de aplicarlo a un chat entre cliente y administrador.

Así que primero se realizó el envío de mensajes, después el despliegue de los mismos de acuerdo al usuario, la actualización del chat, el estatus de usuario para saber si además de estar en línea se encontraba escribiendo y finalmente: Notificar al usuario si tiene mensajes sin leer además de un pequeño anuncio que indica el número de mensajes sin leer.

4.5 Administrador de Personal

Para administrar el personal se desarrolla otra herramienta creada con las técnicas AJAX. En este apartado se construye un apartado donde se pueden introducir empleados a la base de datos, modificar los ya existentes y eliminar los que ya no se encuentren en funciones.

Lo primero que se creó fue el maquetado con ayuda de HTML y de Bootstrap, después una sección en la que se le muestra al usuario una tabla con los empleados que se encuentran dados de alta y un pequeño formulario en el que se pueden ir agregando nuevos elementos o, al darle clic a uno ya existente, mandar a traer los datos del técnico hacia el formulario y editarlos para posteriormente proceder a guardarlos nuevamente en la base de datos.

Aquí en la figura 42 se muestra la vista de la pestaña donde se realizan las actividades antes mencionadas, a donde solamente tiene acceso el usuario administrador, ya que el usuario cliente tiene acceso a otras pestañas muy independientes.



Figura 42. Pestaña del administrador para gestionar personal.

4.5.1 Despliegue De Registros De Personal

La primera función a realizar es mostrartabla(), para poder mandar a imprimir en pantalla los técnicos que están actualmente dados de alta en el sistema. Esta función AJAX de tipo GET manda una petición al documento listamec.php para obtener los registros de la tabla del personal, tal como se aprecia en la figura 43.

```
23 function mostrartabla(){
24     $.ajax({
25         url: 'listamec.php',
26         type: 'GET',
27         success: function(response){
```

Figura 43. Primer fragmento de la función mostrartabla().

El documento listamec.php realiza una consulta de tipo select a la tabla mecánicos de la base de datos, para después dentro de un ciclo while recorrer los registros y los asigna en una variable de tipo JSON para poder devolver en arreglos cada uno de ellos tal como se visualiza en la figura 44.

```
2 include 'conexionagenda.php';
3 $connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
4 $query = "SELECT * from mecanicos";
5 $result = mysqli_query($connection,$query);
6 if (!$result) {
7     die('fallo en la conexion'.mysqli_error($connection));
8 }
9 $json = array();
10 while($row= mysqli_fetch_array($result)){
11     $json[] = array(
12         'nombre' => $row['nombre'],
13         'especialidad' => $row['especialidad'],
14         'id' => $row['idmecanico']
15     );
16 }
17 $jsonstring =json_encode($json);
18 echo "$jsonstring";
```

Figura 44. Consulta de personal en la base de datos.

Después en la función AJAX mostrartabla(), se recibe la respuesta con la información de los registros existentes y, por medio de un maquetado de tabla se mandan a imprimir junto con el botón de eliminar registro, al que más adelante se detallará su funcionamiento, vea la figura 45.

```

let mecanico = JSON.parse(response);
let template = '';
mecanico.forEach(mecanico => {
    template +=
        `<tr mecanic="${mecanico.id}" class="table-dark">
            <td scope="row">
                <a href="#" class="mecan-item" style="color:white;">${mecanico.nombre}
            </td>
            <td scope="row">${mecanico.especialidad}</td>
            <td>
                <button class="eliminar btn btn-danger">
                    Eliminar
                </button>
            </td>
        </tr>`;
});
$('#tasks').html(template);
}

```

Figura 45. Segundo fragmento de la función mostrar tabla.

4.5.2 Registro de Personal

Teniendo lo anterior, se crea la función que agrega nuevos técnicos a la base de datos, la cual hace uso de un pequeño formulario, en el que el administrador debe introducir los datos del nuevo Técnico, y una vez que envía la información, una función de javascript realiza una condicional en la que evalúa el valor de la variable edit. La cual es falsa al momento de entrar a este apartado, y de ser así, envía los datos al documento agregarmec.php. tal como se aprecia en la figura 46. Donde se queda a la espera de la respuesta de la función.

```

7      $('#task-form').submit(function(e) {
8          const postData = {
9              nombre: $('#name').val(),
10             especialidad: $('#description').val(),
11             id: $('#idtar').val()
12         };
13         console.log(postData);
14         let url = edit === false ? 'agregarmec.php' : 'mecanico-edit.php';
15         console.log(url);
16         $.post(url, postData, function(response){
17             console.log(response);
18             mostrartabla();
19             $('#task-form').trigger('reset');
20         });
21         e.preventDefault();
22     });

```

Figura 46. Función condicional.

En la figura 47 se encuentra la Query de inserción de registro a la base de datos, en la que se realiza el llenado de la tabla de mecánicos con su nombre y especialidad. El Id se asigna por default y se va autoincrementando.

```
<?php
include 'conexionagenda.php';
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre)
;
if (isset($_POST['nombre'])) {
    $nombre =$_POST['nombre'];
    $especialidad =$_POST['especialidad'];
    $query = " INSERT into mecanicos(nombre,especialidad) VALUES ('$
        nombre','$especialidad')";
    $result = mysqli_query($connection,$query);
    if (!$result) {
        die('Fallo en la consulta pana....'. mysqli_error($connection))
        ;
    }
    echo "Tarea agregada satisfactoriamente";
}
?>
```

Figura 47. Inserción en la base de datos de nuevo personal.

Al regresar de la Query, se manda a llamar la función mostrartabla() para que se visualice la actualización al momento de realizar la inserción y se resetean los campos del formulario.

Aquí fue donde se vio que sería bueno tener una función que ayude a obtener los datos de un empleado para mandar a traer sus datos al formulario y, una vez ahí, editarlos y actualizarlos, esto en caso de no querer eliminar y solo modificar o realizar correcciones a los técnicos que ya se encuentran dados de alta.

4.5.3 Actualización de Personal

Una vez que ya se tiene un registro, se crea una función que, al dar clic en un registro de la tabla, llene el formulario, en el que se editen los cambios para posteriormente enviarlos de nuevo y actualizar. Todo esto antes mencionado sin tener que actualizar el sitio.

```

85     $(document).on('click', '.mecan-item', function () {
86         let elemento = $(this)[0].parentElement.parentElement;
87         let id = $(elemento).attr('mecanic');
88         $.post('mecanico-simple.php', {id}, function(response){
89             console.log(response);
90             const tarea = JSON.parse(response);
91             $('#name').val(tarea.nombre);
92             $('#description').val(tarea.especialidad);
93             $('#idtar').val(tarea.idmecanico);
94             edit = true;
95         })
96     });
97 });

```

Figura 48. Función que trae a pantalla los datos.

En esta función de la figura 48 se manda al documento mecanico-simple.php el id del registro de la tabla que se seleccionó, esto gracias a la herramienta `$(this)[0].parentElement.parentElement`; (El cero hace referencia a la primera posición en el arreglo, donde se encuentra el ID).

En la figura 49 se visualiza el documento mecanico-simple.php en el que se realiza la consulta que despliega el registro de la tabla de mecánicos, donde coincida el ID que se recibe con alguno de la base de datos. Con ayuda de una variable de tipo JSON se guardan los datos de dicho registro y lo son enviados de vuelta.

```

include 'conexionagenda.php';
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
$id=$_POST['id'];
$query = "SELECT * FROM mecanicos where idmecanico=$id";
$result = mysqli_query($connection,$query);
if (!$result) {
    die('fallo en la edicion de la base'.mysqli_error($connection));
}
$json= array();
while ($row = mysqli_fetch_array($result)) {
    $json[] = array(
        'nombre' => $row['nombre'],
        'especialidad' => $row['especialidad'],
        'idmecanico' => $row['idmecanico']
    );
}
$jsonstring = json_encode($json[0]);
echo $jsonstring;

```

Figura 49. Mecánico-simple.php.

Con esta información, se muestra al administrador el formulario, en el que puede editar los campos o actualizarlos. Mientras que, en segundo plano, la variable edit cambia a true, esto con el fin que al entrar a la función submit, entre al condicional y esta vez sea dirigido al documento mecanico-edit.php, llevando consigo los datos correspondientes para realizar la actualización del sistema, que son el nombre, la descripción tal como se ve en la figura 50.

```
hp
include 'conexionagenda.php';
$connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre);
$id = $_POST['id'];
$nombre = $_POST['nombre'];
$especialidad = $_POST['especialidad'];

$query = "UPDATE mecanicos set nombre='$nombre',especialidad='$especialidad'
WHERE idmecanico='$id'";
$result= mysqli_query($connection,$query);
if(!$result){
    die('Error en la consulta de edicion....'. mysqli_error($connection));
}
echo "Todo al cien con la edicion";
```

Figura 50. Mecánico-edit.php.

Después resetea los campos del formulario y manda a llamar nuevamente a la función mostrar tabla para actualizar las actualizaciones y que el administrador pueda ver si quedó correcto o si necesita modificar algo más.

4.5.4 Eliminación de Personal

Después de esto solo falta una última función, igualmente importante, que es la de eliminar empleados de la base de datos, ya que, cuando por alguna razón estos ya no forman parte del equipo de trabajo, es necesario poder eliminar para que no se cometa un error al asignarle un trabajo futuro.

```

49      $(document).on('click', '.eliminar', function () {
50          let elemento = $(this)[0].parentElement.parentElement;
51          let id = $(elemento).attr('mecanic');
52          Swal.fire({
53              title: '¿Estás seguro?',
54              text: "¡No podrás revertir esto!",
55              icon: 'warning',
56              showCancelButton: true,
57              confirmButtonColor: '#3085d6',
58              cancelButtonColor: '#d33',
59              confirmButtonText: 'Si, aceptar'
60          }).then((result) => {

```

Figura 51. Función eliminar parte uno.

Para empezar, se crea la función eliminar, tal como se aprecia en la figura 51, una función de tipo onclick la cual tiene dentro una alerta del tipo SweetAlert que lanzará un pequeño cuadro de dialogo para preguntar si realmente se quiere eliminar ese registro al que se le ha dado clic. Y de ser así, entonces se procede a mandar el id del registro hacia el documento eliminarmec.php.

```

2      include 'conexionagenda.php';
3      $connection =mysqli_connect($db_host,$db_usuario,$db_contra,$db_nombre)
4      ;
5      if (isset($_POST['id'])) {
6          $id=$_POST['id'];
7          $query = "DELETE from mecanicos WHERE idmecanico=$id";
8          $result = mysqli_query($connection,$query);
9          if(!$result){
10              die('Error en la consulta de eliminacion....'. mysqli_error($
11                  connection));
12          }
13          echo "Tarea eliminada con facilidad";
14      }

```

Figura 52. Query de eliminación.

En la figura 52 se observa cómo se genera la Query que eliminará al empleado de acuerdo al ID recibido y devuelve un echo que manda a imprimir en consola para verificar que realmente haya sido eliminado. Y de no ser así, en la función eliminar se manda otra sweetalert2 con la notificación que el mecánico no ha sido eliminado.

```

62         if (result.isConfirmed) {
63             Swal.fire(
64                 'Eliminado!',
65                 'Este mecánico ha sido eliminado',
66                 'success'
67             )
68             //Aquí vamos a enviar a eliminar, el id de la fila
69             $.post('eliminarmec.php',{id},function (response) {
70                 console.log(response);
71                 mostrartabla();
72             })
73         }else if (
74             result.dismiss === Swal.DismissReason.cancel
75         ){
76             Swal.fire(
77                 'Interrumpido',
78                 'El mecanico está a salvo :)',
79                 'warning'
80             )
81         }
82     })

```

Figura 53. Función eliminar parte 2.

En la figura 53 se visualiza la estructura del sweetalert2 donde se notifica al usuario administrador que el registro sigue a salvo. Con una alerta bastante agradable a la vista tal como se ve en la figura 54.



Figura 54. Sweetalert2.

Así es como está compuesta la gestión del personal con ayuda de AJAX. En donde se puede trabajar con todas las funciones antes mencionadas y facilitar la administración de esta área.

5 Conclusiones y Trabajo Futuro

Se desarrolló la aplicación Coq's Tuning Performance, que mejora la administración y gestión de un taller de servicio automotriz mejorando la atención al cliente mediante el procesamiento de solicitudes en segundo plano por medio de AJAX y el manejo de sesiones. Esta mejora se logra proporcionando una comunicación estrecha entre los clientes y el taller automotriz, así como también agilizando la atención de las solicitudes y citas que se van generando, la respuesta al chat y del manejo del personal y, por supuesto, forjando presencia de forma global en la red de redes. Además, la aplicación Coq's Tuning Performance desarrollada proporciona un entorno sencillo, usable, robusto, seguro, responsivo y en tiempo real; permitiendo actualizaciones constantes sin que el usuario las perciba para que éste se centre sólo en lo que requiere. Este proyecto se desarrolló, iniciando con el análisis de requisitos. Luego el maquetado en html5 puro, después se le dio una mayor ergonomía y diseño gracias al *framework* Bootstrap, posteriormente se realizó la creación de la base de datos, la cual ha ido sufriendo mejoras con las nuevas funciones implementadas; después se programó y finalmente se integró AJAX, donde los requisitos lo demandaban.

Como trabajo futuro se personalizaría la aplicación de acuerdo a las necesidades de cada cliente o lugar de desarrollo, así como desarrollar un Bot, que asista a los clientes las 24 horas del día y los 365 días del año.

6 Bibliografía.

- [1] Sublime Text. Documentation - Sublime Text. (s. f.). Documentación acerca de Sublime Text. Recuperado 19 de septiembre de 2020, de <https://www.sublimetext.com/docs/3/>
- [2] Brackets Manual. (s. f.). CodePen. Recuperado 19 de septiembre de 2020, de <https://codepen.io/gninos/pen/zrNvwZ>
- [3] XAMPP. (s. f.). Manual. xampp-instalacion-windows. Recuperado 19 de septiembre de 2020, de <https://www.mclibre.org/consultar/php/otros/xampp-instalacion-windows.html>
- [4] Yoney Gallardo. (2018, 27 febrero). CURSO COMPLETO DE BOOTSTRAP 4 [Vídeo]. YouTube. <https://www.youtube.com/watch?v=-83eiJ9EaD4>
- [5] Weynman Linda, Weynman William. (2002). Diseño creativo de HTML. Mexico: Pearson education.
- [6] Ralph G Schulz. (2007). Diseño Web con CSS. España: RedLine GMBH.
- [7] Miguel Ángel Arias. (2017). Aprende Programación Web con PHP y MySQL: 2ª Edición. Estados Unidos: IT Campus Academy.
- [8] 000webhost. (s. f.). Características del host gratis - PHP, MySQL, cPanel, sin anuncios. Free Web Host. Recuperado 19 de septiembre de 2020, de <https://mex.000webhost.com/caracteristicas>
- [9] W3School. (s. f.). AJAX Introduction. w3schools.com. Recuperado 19 de septiembre de 2020, de https://www.w3schools.com/xml/ajax_intro.asp
- [10] Jon Duckett, HTML & CSS Design and build Websites. Firth Edition. John Wiley & Sons, Inc. 2011.
- [11] Goldstein, Alexis. (2011). HTML5 y CSS3. Madrid: Anaya Multimedia-Anaya Interactiva.EDIAactive. (2013). Aprender HTML5, CSS3 y Javascript con 100. Barcelona: Marcombo, S.A.Heurtel, olivier. (2019). PHP 7 2ª edición: Desarrollar un sitio web dinámico e interactivo. Barcelona: Ediciones Eni.
- [12] Welling, Luke Thomson, Laura. (2017). Desarrollo Web con PHP y MySQL.5ª Edición. Madrid: Anaya Multimedia-Anaya Interactiva.
- [13] Jacob. (2018). Bootstrap 4 Quick Start: Bootstrap creative.
- [14] Toyota. (s. f.). Toyota MX / Agenda una cita. Toyota Mx. Recuperado 27 de septiembre de 2020, de <https://toyota.mx/agenda-una-cita>
- [15] Honda. (s. f.). 403 Forbidden. Honda Conausa. Recuperado 27 de septiembre de 2020, de <https://www.hondaconausa.com.mx/cita-de-servicio-es-mx.htm#:~:text=Te%20invitamos%20a%20agendar%20una,disponibilidad%20del%20%C3%A1rea%20de%20servicio.>
- [16] Chevrolet. (s. f.). Cita de Servicio | Agenda en línea | Chevrolet México. Chevrolet Mx. Recuperado 27 de septiembre de 2020, de <https://www.chevrolet.com.mx/servicio>

- [17] Nissan. (s. f.). Cita de Servicio. Recuperado 27 de septiembre de 2020, de <https://www.nissan.com.mx/servicio-nissan/cita-de-servicio.html>
- [18] Volkswagen. (s. f.). dccsearch. Volkswagen MX. Recuperado 27 de septiembre de 2020, de <https://www.vw.com.mx/app/concesionarias/vw-mx/esA>
- [19] Ramos, A. (2018, 16 agosto). [jQuery] Instalar y usar el plugin SweetAlert 2. Evilnapsis. <https://evilnapsis.com/2018/08/16/jquery-instalar-y-usar-el-plugin-sweetalert-2/#:%7E:text=El%20plugin%20SweetAlert%20%20sirve,una%20mejor%20impresi%C3%B3n%20al%20usuario.>
- [20] SweetAlert2. (s. f.). A Beautiful, Responsive, Customizable and Accessible (WAI-ARIA) Replacement for JavaScript's Popup Boxes. Recuperado 7 de febrero de 2021, de <https://sweetalert2.github.io/>
- [21] Animate.css | A cross-browser library of CSS animations. (s. f.). Animate. Recuperado 10 de febrero de 2021, de <https://animate.style/>
- [22] Red Hat. (s. f.). ¿Qué es la metodología ágil? Recuperado 18 de junio de 2021, de <https://www.redhat.com/es/devops/what-is-agile-methodology>
- [23] Student Place. (2018, 16 diciembre). Metodologia de Desarrollo de Software. <https://studentplace98.blogspot.com/2018/09/metodologia-de-desarrollo-de-software.html#:~:text=Las%20metodolog%C3%ADas%20cl%C3%A1sicas%20son%20aquellas,se%20culmine%20la%20etapa%20anterior.>