DOCUMENTO DEL ARTICULO DE INVESTIGACION DEL SISTEMA SMARTFIX

Autores:

Cagua Lucio Derek Geanpierre

Delgado Piguave Anthony Paul

Guevara Bustos Yandri David

Iñiga Alvarado Steven Josué

Izquierdo Vallejo Galo Antonio

Mite Solorzano Omar Alan

Resumen

Los talleres de reparación de dispositivos móviles enfrentan problemas operativos críticos debido a la falta de un sistema centralizado. Entre los inconvenientes más comunes se encuentran la duplicidad de registros, errores de facturación, pérdida de información y dificultades para rastrear el estado de las reparaciones.

SmartFix surge como una solución integral basada en una aplicación web que automatiza la gestión del ciclo de reparación, desde la recepción del dispositivo hasta la entrega, incluyendo el control de inventario, la administración de técnicos y la generación de reportes analíticos. Este sistema permite mejorar la eficiencia, reducir errores y brindar una experiencia más satisfactoria al cliente.

Este artículo presenta la arquitectura del sistema, los requisitos funcionales, los módulos clave, las perspectivas futuras y las estrategias de mitigación de riesgos. El objetivo es demostrar cómo una solución digital puede optimizar significativamente los procesos en un taller técnico, mejorando la productividad, la trazabilidad y la calidad del servicio.

Palabras clave: Servicio técnico, Reparación celular, Gestión de inventario, Automatización, Plataforma web, SmartFix.

Abstract

Mobile device repair workshops often suffer from operational inefficiencies due to the lack of an integrated system. Common issues include duplicate records, billing errors, lost data, and difficulty tracking repair statuses.

SmartFix is a comprehensive web-based solution that automates the entire repair cycle—from device reception to delivery—while managing inventory, technician assignments, and generating analytical reports. This system improves operational efficiency, reduces errors, and enhances customer satisfaction.

This article presents the system architecture, functional requirements, key modules, future enhancements, and risk mitigation strategies. The goal is to demonstrate how a digital platform can streamline repair workshop operations, improving productivity, traceability, and service quality.

Keywords: Technical service, Mobile repair, Inventory management, Automation, Web platform, SmartFix.

Introducción

En la era digital, los talleres de reparación celular requieren soluciones tecnológicas que les permitan ofrecer un servicio eficiente, confiable y competitivo. SmartFix responde a esta necesidad mediante una plataforma web que automatiza los procesos operativos del taller.

Actualmente, muchos talleres dependen de métodos manuales como hojas de cálculo o libretas, lo cual genera errores frecuentes, tiempos de respuesta lentos y escasa visibilidad sobre los datos operativos. Esta situación repercute negativamente tanto en la gestión interna como en la experiencia del cliente.

Este proyecto, desarrollado por estudiantes de la Universidad de Guayaquil en la carrera de Ingeniería en Software, busca implementar un sistema robusto que centralice toda la operación del taller: desde el registro del cliente hasta la entrega del equipo, incluyendo inventario, técnicos, servicios y facturación.

Objetivos

1. Desarrollar un sistema de gestión integral para talleres de reparación celular, enfocado en la automatización del ciclo de servicio.
2. Mejorar la eficiencia en la administración de clientes, técnicos, inventario y órdenes de reparación.
3. Minimizar errores manuales y duplicidad de información mediante validaciones automáticas.
4. Proporcionar reportes analíticos que ayuden en la toma de decisiones estratégicas.
5. Establecer una arquitectura escalable, confiable y con trazabilidad completa de las operaciones.

Metodología

El desarrollo del sistema SmartFix se llevó a cabo mediante un enfoque iterativo y modular, estructurado en fases que abarcaron desde el levantamiento de requerimientos hasta las pruebas e implementación del sistema. Se emplearon buenas prácticas de ingeniería de software con una organización por entregables y validaciones constantes.

La construcción del sistema siguió los siguientes pasos:

- **Análisis de requerimientos**: Se identificaron las necesidades operativas de los talleres de reparación, definiendo los actores principales del sistema (cliente, técnico, administrador) y sus respectivos flujos de interacción.
- **Diseño detallado**: Se elaboraron los diagramas de casos de uso, modelo de clases y arquitectura del sistema. La base de datos fue modelada en MySQL Workbench, considerando la integridad referencial y la escalabilidad.
- **Desarrollo modular**: Cada módulo (Clientes, Técnicos, Productos, Servicios, Ordenes, Facturación) fue implementado de manera independiente en PHP y conectado a la base de datos MySQL a través de formularios web responsivos.
- **Control de versiones**: Se utilizó Git como sistema de control de versiones, lo que permitió realizar seguimiento de cambios y facilitar la colaboración del equipo de desarrollo.
- **Pruebas sistemáticas**: Se definió un plan de pruebas que incluyó pruebas funcionales, de regresión, de rendimiento, seguridad y de instalación. Las herramientas utilizadas fueron PHPUnit, Jest y Selenium WebDriver, con resultados documentados en protocolos de prueba.
- **Despliegue y validación final**: El sistema fue instalado en un entorno remoto mediante el uso de XAMPP y FileZilla, asegurando su operatividad desde un dominio público.

Estimación COCOMO

COCOMO Básico – proyecto orgánico

Submodelos básicos	a	b	c	d
Orgánico	2,4	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	3,6	1,20	2,5	0,32

1. SUBMODELOS DE COCOMO I

SIZE Estimado = 16.15 KLOC

1.1. ORGÁNICO

Esfuerzo = $Effort = a * (SIZE)^b$ = 2.4 * (16.15)$^{1.05}$ = 44.5 PM (Personas-Mes)

Tiempo de Desarrollo = $DevTime = c * (Effort)^d$ = 2.5 * (44.5)$^{0.38}$ = 10.6 M (Meses)

=> **Personas**: 44.5/10.6 = 4.2 personas

=> **Productividad**: 16150/44.5 = 362.92 LOC/PM

1.2. SEMI-ACOPLADO

Esfuerzo = $Effort = a * (SIZE)^b$ = 3.0 * (16.15)$^{1.12}$ = 67.7 PM (Personas-Mes)

Tiempo de Desarrollo = $DevTime = c * (Effort)^d$ = 2.5 * (67.7)$^{0.35}$ = 10.9 M (Meses)

=> **Personas**: 67.7/10.9 = 6.21 personas

=> **Productividad**: 16150/67.7 = 238.55 LOC/PM

1.3. EMPOTRADO

Esfuerzo = $Effort = a * (SIZE)^b$ = 3.6 * (16.15)$^{1.20}$ = 101.42 PM (Personas-Mes)

DevTime = $DevTime = c * (Effort)^d$ = 2.5 * (101.42)$^{0.32}$ = 10.96 M (Meses)

=> **Personas**: 101.42/10.96 = 9.25 personas

=> **Productividad**: 16150/101.42 = 159.24 LOC/PM

Para la aplicación web el tipo **orgánico** es el más adecuado a usar:

- Es el que menos esfuerzo requiere.
- Tiene la mayor productividad.
- Implica un equipo pequeño y estable

2. COCOMO Intermedio – Proyecto Orgánico

Submodelos intermedios	a	b	c	d
Orgánico	3,2	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	2,8	1,20	2,5	0,32

2.1. Tabla Factores de Costo

Atributos	Valor					
	Muy bajo	Bajo	Normal	Alto	Muy Alto	Extra alto
Atributos de Software						
Fiabilidad	0,75	0,88	1	1,15	1,4	
Tamaño de Base de datos		0,94	1	1,08	1,16	
Complejidad	0,7	0,85	1	1,15	1,3	1,65
Atributos de hardware						
Restricciones de tiempo de ejecución			1	1,11	1,3	1,66
Restricciones de memoria virtual			1	1,06	1,21	1,56
Volatilidad de la máquina virtual		0,87	1	1,15	1,3	
Tiempo de respuesta		0,87	1	1,15	1,3	
Atributo de personal						
Capacidad de análisis	1,46	1,19	1	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1	0,91	0,82	
Calidad de los programadores	1,42	1,17	1	0,86	0,7	
Experiencia en la máquina virtual	1,21	1,1	1	0,9		
Experiencia en lenguaje	1,14	1,07	1	0,95		
Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,1	1	0,91	0,82	
Utilización de herramientas de software	1,24	1,1	1	0,91	0,83	
Restricciones de tiempo de desarrollo	1,23	1,08	1	1,04	1,1	

$$\sum factores\ funcionales\ de\ costo$$
$$= 1 + 1.8 + 1.15 + 1.11 + 1.06 + 0.87 + 1.15 + 1 + 1.13 + 1 + 1 + 1 + 1 + 1 + 1.08$$
$$= \mathbf{16.35}$$

2.2. Cálculo del Size

	Factores de Peso				
Factores Funcionales de Peso	Parámetros de Medida (1)			Contador (2)	Total Multiplicación (1) *(2)
	Simple	Media	Compleja		
N.º Entrada de usuario	7	10	15	20	200
N.º Salida usuario	5	7	10	10	70
N.º Consulta usuario	3	4	6	8	32
N.º Archivos Lógicos Internos (tablas)	4	5	7	4	20
N.º Interfaces externas	3	4	6	6	24
			$\sum$ Factores de Peso =		346

$$PF = \left[\sum factores\ funcionales\ de\ peso\right] * \left[0.65 + \left(0.01 + \sum factores\ funcionales\ de\ costo\right)\right]$$

$$PF = [346] * [0.65 + (0.01 * 16.35)] = \mathbf{281.471}$$

$$PF = \mathbf{281.471}$$

2.2.1. Calculo formulaLOC

Tabla de conversión de: Correlación código fuente a PF

Lenguaje	Correlación Código Fuente por PF (aprox)
Assembler	320
C	128
ALGOL	105
FORTRAN	105
PASCAL	91
RPG	80
PL/1	80
Modula-2	80
Prolog	64
LISP	64
BASIC	64
4GL para BD	40
APL	32
Smaltallk	29
Query	13
Spreadsheet	6
Sql	13
VB	24
Java	46
Html	14
Delphi	118
C++	53
COBOL	107
C#	58
PHP	50

$$LOC = PF * Correlación$$

$$LOC = 281.471 * 50$$

$$LOC = 14073.55$$

$$KLOC = 14073.55/1000$$

$$\boldsymbol{KLOC = 14.073}$$

2.3. Cálculo de la Variable FAE (multiplicador)

Atributos	Valor					
	Muy bajo	Bajo	Normal	Alto	Muy Alto	Extra alto
Atributos de Software						
Fiabilidad	0,75	0,88	1	1,15	1,4	
Tamaño de Base de datos		0,94	1	1,08	1,16	
Complejidad	0,7	0,85	1	1,15	1,3	1,65
Atributos de hardware						
Restricciones de tiempo de ejecución			1	1,11	1,3	1,66
Restricciones de memoria virtual			1	1,06	1,21	1,56
Volatilidad de la máquina virtual		0,87	1	1,15	1,3	
Tiempo de respuesta		0,87	1	1,15	1,3	
Atributo de personal						
Capacidad de análisis	1,46	1,19	1	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1	0,91	0,82	
Calidad de los programadores	1,42	1,17	1	0,86	0,7	
Experiencia en la máquina virtual	1,21	1,1	1	0,9		
Experiencia en lenguaje	1,14	1,07	1	0,95		
Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,1	1	0,91	0,82	
Utilización de herramientas de software	1,24	1,1	1	0,91	0,83	
Restricciones de tiempo de desarrollo	1,23	1,08	1	1,04	1,1	

$FAE = 1 * 1.8 * 1.15 * 1.11 * 1.06 * 0.87 * 1.15 * 1 * 1.13 * 1 * 1 * 1 * 1 * 1 * 1.08 =$ **1.949521295**

$$\boldsymbol{Esfuerzo(E) = a(KLOC)^b * FAE}$$

$$\boldsymbol{Duración\ (D) = c(E)^d}$$

$$\boldsymbol{Personal\ (P) = E/D}$$

Submodelos intermedios	a	b	c	d
Orgánico	3,2	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	2,8	1,20	2,5	0,32

ENTONCES SE TIENE:

$E = 3{,}0(\mathbf{14.073})^{1{,}12} * 1.949 = 113.012\ \ meses/\ hombre$

$D = 2{,}5(113.012)^{0{,}35} = \ 13{,}077\ meses$

$P = \frac{113.012}{13{,}077} = 8.64$

$C = 8.64 * 1000 = 8640$

COCOMO II

2.4. Datos iniciales

A = 2.5
SIZE = 14.073 KLOC (14.073 líneas de código)

Escalamiento B:

- Factores de escala dados:
  - PREC = nominal (3.72)
  - FLEX = very high (1.01)
  - RESL = high (2.83)
  - TEAM = extra high (0.00)
  - PMAT = nominal (4.68)

Multiplicadores de Esfuerzo (M):

- PERS = high (0.83)

Multiplicadores M

Cost Drivers	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
RCPX	0.73	0.81	0.98	1.0	1.30	1.74	2.38
RUSE	-	-	0.95	1.0	1.07	1.15	1.24
PDIF	-	-	0.87	1.0	1.29	1.81	2.61
PERS	2.12	1.62	1.26	1.0	0.83	0.63	0.50
PREX	1.59	1.33	1.12	1.0	0.87	0.71	0.62
FCIL	1.43	1.30	1.10	1.0	0.97	0.73	0.62
SCED	-	1.43	1.14	1.0	1.0	1.0	-

$$\mathbf{B} = 0.91 + 0.01 * (3.72 + 1.01 + 2.83 + 0.00 + 4.68)$$

$$\mathbf{B} = 0.91 + 0.01 * (12.24)$$

$$\mathbf{B} = \mathbf{1.0324}$$

$$\mathbf{PM} = 2.5 * 14.073^{1.0324}$$

$$\mathbf{PM} = \mathbf{38.329}\ \textbf{Person Months}$$

$$\mathbf{E} = 38.329 * (0.83)$$

$$\mathbf{E} = \mathbf{31.813}\ \textbf{Person Months}$$

5. MODELO POST ARQUITECTURA

$$PM_{adjusted} = PM_{nominal} * \prod_{i=7}^{17} EM_i$$

$$TDEV_{nominal} = \left[\emptyset * \left(PM_{adjusted}\right)^{(0.28+0.2(B-0.091))}\right] * \frac{SCED\%}{100}$$

$\boldsymbol{B} = 1.0324$

$\boldsymbol{PM} = 38.329$

$\emptyset = 3.67$

Cost Drivers:

RELY = Low (0.75)

TIME = Very high (1.11)

ACAP = Very low (1.22)

PCAP = Very low (1.16)

TOOL = Low (0.86)

SCED = Very low (1.10)

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.75	0.88	1.00	2.48	1.24	0.00
DATA		0.93	1.00	2.03	2.03	0.00
CPLX	0.75	0.88	1.00	2.83	1.41	0.00
RUSE		0.91	1.00	2.19	1.10	0.00
DOCU	0.89	0.95	1.00	3.12	1.56	0.00
TIME			1.00	1.11	1.31	1.67
STOR			1.00	1.06	1.21	1.57
PVOL		0.87	1.00	1.15	1.30	
ACAP	1.50	1.22	1.00	0.83	0.67	
PCAP	1.37	1.16	1.00	0.87	0.74	
PCON	1.24	1.10	1.00	0.92	0.84	
AEXP	1.22	1.10	1.00	0.89	0.81	
PEXP	1.25	1.12	1.00	0.88	0.81	
LTEX	1.22	1.10	1.00	0.91	0.84	
TOOL	1.24	1.12	1.00	0.86	0.72	
SITE	1.25	1.10	1.00	0.82	0.84	0.78
SCED	1.29	1.10	1.00	1.00	1.00	

$$PM_{adjusted} = 38.329 * (0.75 * 1.11 * 1.22 * 1.16 * 0.86 * 1.10)$$

$$= 38.329 * (1.114)$$

$$= \mathbf{42.698\ Person\ Months}$$

$$TDEV_{nominal} = \left[3.67 * (42.698)^{(0.28+0.2(1.0324-0.091))}\right] * \frac{1.10}{100}$$

$$= \left[3.67 * (42.698)^{(0.46828)}\right] * \frac{1.10}{100}$$

$$= [3.67 * 5.8007] * \frac{1.10}{100}$$

$$= [21.286] * \frac{1.10}{100}$$

$$= \mathbf{0.234\ Meses}$$

Metodología de Desarrollo

Herramientas para el desarrollo del sistema

XAMPP

XAMPP es un entorno de desarrollo local que incluye Apache, PHP y MySQL. Permitió simular un servidor web real durante la fase de desarrollo y pruebas del sistema SmartFix. Su facilidad de configuración y portabilidad contribuyó a una implementación eficiente de los módulos en un entorno controlado.

MySQL Workbench

MySQL Workbench fue la herramienta utilizada para modelar y administrar la base de datos. Su interfaz gráfica permitió diseñar esquemas relacionales, establecer integridad referencial y realizar consultas complejas de forma intuitiva. Facilitó además la documentación visual del modelo de datos del sistema.

PHP y JavaScript

El sistema fue desarrollado principalmente en PHP para la lógica del lado del servidor y JavaScript para mejorar la interacción en el cliente. Estos lenguajes ofrecen amplia compatibilidad con entornos web, una curva de aprendizaje accesible y gran soporte de la comunidad. Permiten el desarrollo ágil de interfaces dinámicas y funcionales.

Git

Git fue utilizado como sistema de control de versiones, permitiendo un seguimiento detallado de los cambios realizados en el código fuente. Facilitó la colaboración entre los integrantes del equipo, el manejo de ramas y la integración continua de nuevas funcionalidades.

Chrome DevTools

Esta herramienta de desarrollo del navegador Google Chrome permitió inspeccionar elementos de la interfaz, analizar el rendimiento y depurar errores en tiempo real. Resultó clave para garantizar una experiencia de usuario fluida y libre de errores visuales.

Selenium WebDriver

Selenium se utilizó para automatizar pruebas funcionales en el navegador, simulando la interacción de un usuario real con el sistema. Esto ayudó a validar flujos críticos como el registro de clientes o la gestión de productos.

PHPUnit y Jest

PHPUnit fue usado para realizar pruebas unitarias en el backend (PHP), mientras que Jest se utilizó para probar funciones JavaScript en el frontend. Ambas herramientas permitieron detectar errores en etapas tempranas y asegurar la estabilidad del código y distribución de roles entre los integrantes

Visión del proyecto

- **Objetivo**: Desarrollar un sistema web para la gestión integral de talleres técnicos, permitiendo administrar procesos como recepción de equipos, registro de órdenes, control de inventario y atención al cliente.

Planificación del proyecto

- Lista de funcionalidades: Se elaboró un conjunto de requerimientos organizados por módulos, incluyendo clientes, técnicos, productos, servicios y control de acceso.

- Actores clave: Se consideraron los perfiles de administrador, técnico y cliente para estructurar los casos de uso y flujos de trabajo.

- Organización del equipo: Se asignaron roles funcionales como responsable técnico, líder del proyecto, desarrolladores y encargado de documentación.

Etapas de desarrollo

- Preparación inicial: Se configuró el entorno de trabajo, se diseñó la base de datos y se definieron objetivos por módulo.

- Desarrollo modular: Las funcionalidades se implementaron de forma progresiva, permitiendo validar y corregir errores en cada fase.

- **Validaciones constantes**: Se realizaron pruebas frecuentes y revisiones entre miembros del equipo para asegurar la calidad y el cumplimiento de requisitos.

Entrega y revisión del sistema

- Validación final: Al finalizar el desarrollo se evaluó el sistema en conjunto, verificando el cumplimiento de funcionalidades y estabilidad general.

- **Instalación remota**: El sistema fue desplegado en un servidor público mediante XAMPP y FileZilla, quedando listo para uso real.

Mantenimiento y mejora continua

- **Soporte técnico**: Se previó un periodo post-despliegue para corregir fallos menores.

- **Lecciones aprendidas**: Se documentó el proceso para mejorar la planificación en futuros proyectos.

Distribución de roles entre los integrantes

Nombres	Contacto	Rol
Cagua Lucio Derek Geanpierre	derek.cagualuc@ug.edu.ec	Responsable de Documentación Técnica
Delgado Piguave Anthony Paul	anthony.delgadopig@ug.edu.ec	Desarrollador Front-End
Guevara Bustos Yandri David	yandridavid@hotmail.com	Desarrollador Front-End
Iñiga Alvarado Steven Josué	steven.inigaa@ug.edu.ec	Responsable de Documentación Técnica
Izquierdo Vallejo Galo Antonio	galo.izquierdov@ug.edu.ec	Desarrollador Back-End
Mite Solórzano Omar Alan	alan.mitesol@ug.edu.ec	Responsable de Documentación Técnica

Resultado

Para el proyecto SmartFix, el resultado esperado fue la creación de un sistema de gestión integral para talleres técnicos de reparación celular, que permita automatizar los procesos operativos, mejorar la trazabilidad de las órdenes de servicio y aumentar la eficiencia del personal. El sistema desarrollado cumple con estos objetivos mediante una estructura clara, una base de datos bien diseñada y una interfaz funcional adaptable.

Objetivos y Funcionalidades del Sistema:

- **Gestión integral del taller**: El sistema permite registrar y administrar clientes, técnicos, productos y servicios de forma centralizada.
- **Control de inventario**: Administra el stock de repuestos y accesorios, permitiendo actualizaciones de ingreso, salida y edición de productos.
- **Administración de órdenes de servicio**: Facilita el seguimiento del estado de cada reparación, asignación de técnicos y control de entregas.
- **Reportes y análisis**: Genera reportes estratégicos que permiten evaluar el desempeño de los servicios y el movimiento del inventario.
- **Interfaz intuitiva y validaciones**: Presenta una interfaz clara y amigable, con validaciones que garantizan la integridad de los datos y alertas para mejorar la experiencia del usuario.

Arquitectura:

El sistema SmartFix implementa una arquitectura basada en **componentes funcionales distribuidos**, similar al modelo **Datos-Entidad-Negocio-Presentación (DNP)**, adaptada a entornos web. Esta estructura favorece la organización del sistema, la separación de responsabilidades y la escalabilidad del software. Se describen a continuación los componentes:

- **Datos**: Gestiona el acceso a la base de datos MySQL. Aquí se concentran las operaciones de lectura, escritura, actualización y eliminación de datos, así como el uso de claves primarias y foráneas para garantizar la integridad referencial.
- **Entidad**: Incluye las clases que representan las entidades del sistema como Cliente, Técnico, Producto y Servicio. Estas estructuras encapsulan los atributos del dominio, manteniendo independencia respecto a la lógica de negocio y a la interfaz.
- **Negocio**: Contiene la lógica central del sistema, encargada de procesar las solicitudes del usuario, aplicar reglas, validar datos y coordinar el acceso a la información. Esta capa fue implementada principalmente en PHP.
- **Presentación**: Corresponde a la interfaz del sistema, compuesta por formularios responsivos y un menú de navegación lateral. Brinda acceso a las funcionalidades del sistema mediante una experiencia de usuario clara, validaciones visuales y retroalimentación inmediata.

Calidad del Sistema:

Extensibilidad:

- La arquitectura modular basada en el modelo Datos–Entidad–Negocio–Presentación (DNP) permite la incorporación de nuevas funcionalidades sin afectar los componentes existentes del sistema.

Confiabilidad:

- La separación de capas garantiza que procesos críticos como el registro de órdenes, la gestión de inventario o la generación de reportes funcionen correctamente, incluso ante actualizaciones de otras áreas del sistema.

Portabilidad:

- Al estar construido con tecnologías multiplataforma como PHP, MySQL y navegadores web modernos, el sistema puede ser trasladado o adaptado fácilmente a diferentes entornos técnicos, tanto locales como remotos.

Requisitos de Hardware y Software:

El sistema especifica los requisitos mínimos necesarios para su correcto funcionamiento, asegurando compatibilidad con las plataformas y el hardware del entorno de trabajo técnico.

Conclusión

Se ha desarrollado una solución de software eficiente y robusta basada en una arquitectura modular tipo Datos–Entidad–Negocio–Presentación (DNP), orientada a la gestión integral de talleres técnicos de reparación celular. El objetivo principal del proyecto fue automatizar procesos como el registro de clientes, gestión de órdenes, administración de inventario y seguimiento de servicios, optimizando así la operatividad del taller y mejorando la experiencia tanto del personal técnico como del cliente final.

La solución implementada permite gestionar datos de clientes, técnicos y productos con precisión, eliminando errores propios de registros manuales y facilitando la trazabilidad de cada servicio técnico. Asimismo, el sistema incluye funcionalidades clave como generación de reportes, validaciones automáticas, control de accesos y una interfaz web clara y accesible.

El sistema ha sido diseñado pensando en los distintos perfiles de usuario que interactúan con él, permitiendo que cada uno acceda a las funciones necesarias según su rol. Su implementación garantiza una experiencia confiable y adaptable, reduciendo tiempos de operación, mejorando la productividad del personal y fortaleciendo la gestión administrativa del taller técnico.