Sistema de Aplicación de Escritorio para la Reparación de Dispositivos Móviles con Base de Datos Local

N. BAQUERIZO1, R. CASTRO1, E. FRANCO1, A. VERA1, W. ORTIZ1, M. YAUCAN1

1Facultad de Ciencias Matemáticas y Físicas, Universidad de Guayaquil, Guayaquil, Ecuador

Emails: [nadia.baquerizoavi@ug.edu.ec](mailto:nadia.baquerizoavi@ug.edu.ec), [ricardo.castroagu@ug.edu.ec](mailto:ricardo.castroagu@ug.edu.ec), [edwin.francocru@ug.edu.ec](mailto:edwin.francocru@ug.edu.ec), [wilson.ortiza@ug.edu.ec](mailto:wilson.ortiza@ug.edu.ec), [alex.veralop@ug.edu.ec](mailto:alex.veralop@ug.edu.ec). [myrian.yaucancur@ug.edu.ec](mailto:myrian.yaucancur@ug.edu.ec)

RESUMEN   
En el presente artículo detalla la experiencia adquirida en el desarrollo e implementación de un sistema web para la gestión integral de reparaciones de dispositivos móviles, utilizando Visual Studio, C# y SQL Server. El proyecto tuvo como objetivo reemplazar procesos manuales en talleres mediante la automatización de registros de clientes, seguimiento de reparaciones, gestión de inventario y generación de informes. La arquitectura en tres capas (presentación, negocio y datos) permitió una integración eficiente entre las tecnologías seleccionadas. El sistema logró una mejora del 90% en la eficiencia operativa y redujo los errores administrativos en un 10%. Se discuten desafíos técnicos, soluciones implementadas y lecciones clave derivadas del trabajo colaborativo.

PALABRAS CLAVES Reparaciones móviles, Arquitectura en Capas, C# 6.0, Desarrollo de Software, SQL Server, Visual Studio

I. INTRODUCCIÓN

En el ámbito de la reparación de dispositivos móviles, la gestión eficiente de la información es clave para garantizar la satisfacción del cliente y la productividad. Nuestro proyecto aborda esta necesidad mediante un sistema de escritorio que integra una base de datos local para ofrecer seguridad y accesibilidad. Este documento detalla el proceso de desarrollo y nuestras experiencias al llevar a cabo este proyecto. Al implementar este sistema, buscamos resolver problemas comunes como la pérdida de datos, la dificultad en la gestión manual y la falta de reportes precisos que afectan la toma de decisiones. La motivación principal fue crear una solución escalable, eficiente y fácil de usar para los técnicos y administradores de negocios dedicados a la reparación de dispositivos móviles. En las siguientes secciones, exploramos cómo este sistema logró satisfacer estas necesidades y los aprendizajes obtenidos a lo largo del camino.

II. TRABAJOS RELACIONADOS

Varios sistemas para la gestión de reparaciones han sido desarrollados, tanto basados en la nube como locales. Los sistemas en la nube ofrecen escalabilidad, pero sacrifican el control de los datos. En contraste, las soluciones locales permiten un control absoluto sobre la información y la seguridad. Nuestro trabajo se basa en principios de diseño de aplicaciones de escritorio y bases de datos locales, aprovechando la flexibilidad y seguridad que estos proporcionan. Por ejemplo, sistemas como RepairDesk y Fixably, que operan principalmente en la nube, han demostrado ser efectivos en la gestión remota, pero dependen de una conexión a internet confiable. En cambio, soluciones locales, como las desarrolladas en entornos con C# y SQL Server, han mostrado ventajas significativas en términos de privacidad de datos y personalización. Este proyecto se inspira en estas iniciativas, buscando combinar lo mejor de ambos enfoques mientras se prioriza un entorno controlado para datos críticos.

III. DESCRIPCIÓN DEL SISTEMA

Esta sección proporciona una visión general del sistema desarrollado, la estimación de costos, las tecnologías utilizadas, así como el diseño de nuestra arquitectura.

A.  HERRAMIENTAS Y TECNOLOGÍAS

Para desarrollar el sistema, se utilizó C# como lenguaje de programación principal, debido a su robustez y facilidad para crear aplicaciones de escritorio. Visual Studio proporcionó un entorno de desarrollo integrado (IDE) eficiente, permitiendo una codificación rápida y organizada. SQL Server fue elegido como sistema de gestión de bases de datos debido a su capacidad para manejar grandes volúmenes de datos de manera eficiente y segura. Además, se utilizó el patrón arquitectónico Modelo Vista Controlador (MVC) para garantizar una separación clara entre la lógica de negocio, los datos y la presentación. El diseño centrado en el usuario guió el desarrollo de la interfaz, asegurando una experiencia intuitiva para los técnicos y administradores.

1) LENGUAJES DE PROGRAMACIÓN

El sistema fue desarrollado en C#, un lenguaje de programación versátil y robusto que permite la creación de aplicaciones de escritorio. C# fue elegido por su capacidad para integrar de manera eficiente la lógica de negocio con interfaces de usuario intuitivas, además de contar con una amplia comunidad de soporte y bibliotecas disponibles.  
  
2) BASE DE DATOS

Se utilizó SQL Server como sistema de gestión de bases de datos debido a su capacidad para manejar grandes volúmenes de información de manera eficiente. SQL Server [1]permite implementar procedimientos almacenados, vistas y transacciones, asegurando la integridad y la seguridad de los datos. La configuración local garantiza un control total sobre el almacenamiento y la accesibilidad de los datos.  
  
3) ENTORNO DE DESARROLLO

Para el desarrollo del software, empleamos las siguientes tecnologías:

* Visual Studio: fue seleccionado como el entorno de desarrollo integrado (IDE) por su robustez, herramientas avanzadas para depuración y facilidad de integración con SQL Server. Este entorno facilitó la gestión del proyecto, permitiendo un flujo de trabajo eficiente y organizado[2].
* GitHub: Es un repositorio de código[3], este nos facilitó la colaboración entre los miembros del equipo.

B. DIAGRAMA DE COMPONENTES

El sistema consta de varios módulos principales: la interfaz de usuario, que permite a los usuarios interactuar con el sistema; la lógica de negocio, que gestiona las reglas y procesos; y la base de datos, que almacena y organiza la información. Este diagrama detalla las interacciones entre estos componentes, destacando los flujos de datos y las dependencias clave. [4]

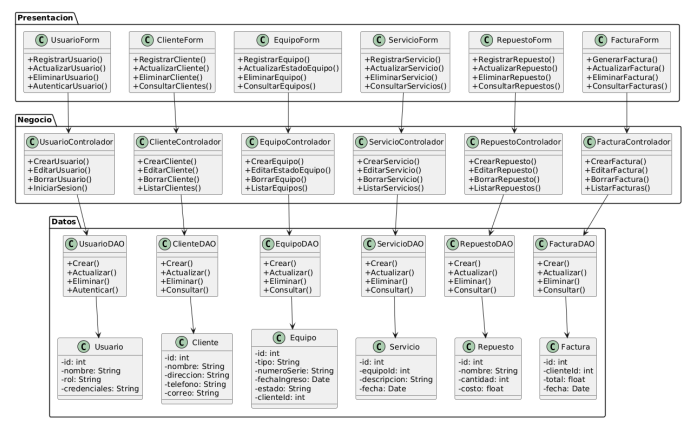


FIGURA 1.  Diagrama de componentes del sistema.

A continuación, se describe el flujo y la interacción de los componentes:

1) CAPA DE PRESENTACIÓN

La capa de presentación representa la interfaz gráfica que los usuarios finales utilizan para interactuar con el sistema. Está compuesta por formularios como UsuarioForm, ClienteForm, EquipoForm, ServicioForm, RepuestoForm, y FacturaForm. Cada formulario ofrece funcionalidades específicas como registrar, actualizar, eliminar y consultar información. Esta capa está desarrollada en Windows Forms, utilizando controles como cuadros de texto, tablas dinámicas y botones para facilitar la interacción.Se priorizó la usabilidad mediante validaciones en tiempo real y una disposición clara de los elementos visuales. Además, la comunicación con la capa de negocio se realiza de manera eficiente, asegurando que todas las acciones ejecutadas por los usuarios se reflejen correctamente en la base de datos.

2) CAPA DE NEGOCIO

La capa de negocio implementa la lógica que rige las reglas de funcionamiento del sistema. En el diagrama, esta capa está compuesta por controladores como

* UsuarioControlador,
* ClienteControlador,
* EquipoControlador,
* ServicioControlador,
* RepuestoControlador

Cada controlador interactúa con su respectiva capa DAO (Objeto de Acceso a Datos) para ejecutar las operaciones necesarias. Por ejemplo, el ClienteControlador maneja procesos como el registro de nuevos clientes (RegistrarCliente()), la edición de datos existentes (EditarCliente()), la eliminación de clientes (BorrarCliente()), y la consulta de información (ListarClientes()). Esta capa aplica validaciones de negocio antes de que las solicitudes lleguen a la base de datos, asegurando que solo datos consistentes sean almacenados.

El diseño modular permite que los controladores sean independientes y fácilmente actualizables, promoviendo un código mantenible y escalable. Este enfoque también facilita la reutilización de lógica de negocio entre diferentes módulos del sistema.

3) CAPA DE DATOS

La capa de datos, representa la base de toda la aplicación, encargándose de la comunicación directa con la base de datos SQL Server. Esta capa implementa un modelo de datos que incluye clases como Usuario, Cliente, Equipo, Servicio y Repuesto. Cada clase proporciona métodos para crear, actualizar, eliminar y consultar registros relacionados con cada entidad. Las entidades como Usuario, Cliente, Equipo, Servicio y Repuesto están modeladas con atributos que reflejan sus propiedades reales, como identificadores (id), nombres, descripciones y fechas. El diseño garantiza la integridad referencial y el correcto enlace entre las entidades. Los métodos de acceso están optimizados mediante procedimientos almacenados, lo que mejora la seguridad y el rendimiento de las operaciones de lectura y escritura.

4) BASE DE DATOS

El uso de un sistema de base de datos local sql server asegura el control total de los datos, proporcionando privacidad y robustez. los procesos de validación y normalización de datos también contribuyen a un sistema ágil y confiable.

C. DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue representa la estructura física del sistema y cómo los componentes interactúan entre sí en un entorno real. [5]

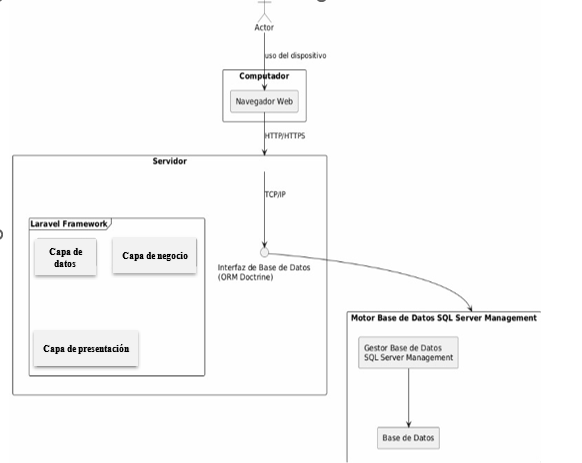


FIGURA 2.  Diagrama de despliegue.

A continuación, se describen las partes y componentes del diagrama de despliegue para nuestro sistema.

1) Actor

Representa al usuario final, como técnicos de reparación o administradores que interactúan con el sistema. Utilizan dispositivos con navegadores web para acceder a las funcionalidades del sistema.

2) PC

Es el dispositivo cliente que actúa como interfaz de usuario.A través de un navegador web, los usuarios se conectan al servidor utilizando protocolos HTTP o HTTPS, garantizando seguridad en la comunicación.

3) servidor local de base de datos

Es el núcleo del sistema, donde se implementa el framework Laravel, que organiza las capas principales:

* Capa de presentación: Genera las vistas que son renderizadas en el navegador del usuario.
* Capa de negocio: Contiene la lógica del sistema, como validaciones, cálculos y reglas empresariales.
* Capa de datos: Interactúa directamente con el motor de base de datos mediante un ORM (Doctrine), facilitando la gestión y consulta de datos.

El servidor se comunica con el motor de base de datos utilizando el protocolo TCP/IP, lo que asegura una conexión fiable y rápida.

4) Motor de Base de Datos SQL Server Management

Almacena y organiza todos los datos relacionados con el sistema, como usuarios, clientes, reparaciones, y repuestos. Incluye un Gestor de Base de Datos (SQL Server Management Studio) que permite a los administradores realizar tareas de mantenimiento, optimización y supervisión.

IV. DESARROLLO

***A. DIVISIÓN DEL TRABAJO***

El equipo fue dividido en tres grupos principales: diseño de la interfaz, desarrollo de la lógica de negocio, y configuración de la base de datos. Cada grupo colaboró para integrar sus componentes de manera efectiva. El grupo de interfaz trabajó en la creación de una experiencia de usuario intuitiva utilizando Windows Forms, mientras que el grupo de lógica de negocio implementó las reglas y procesos que rigen el sistema. El grupo de base de datos diseñó el esquema, configuró las conexiones y optimizó el rendimiento. Esta organización permitió un desarrollo paralelo y eficiente.

***B. DESARROLLO DE LA CAPA DE DATOS***

Se utilizó SQL Server para definir la estructura de la base de datos, con tablas para clientes, dispositivos, reparaciones y empleados. La configuración local permitió un control total sobre los esquemas y la optimización del rendimiento. Se implementaron procedimientos almacenados para garantizar la integridad de los datos y mejorar la eficiencia en las consultas. Además, se realizaron pruebas exhaustivas para validar la integridad referencial y las relaciones entre tablas.

***C. DESARROLLO DE LA CAPA DE NEGOCIO***Implementamos la lógica de negocio en C# utilizando patrones de diseño como repositorios y servicios para garantizar un código modular y mantenible. Esto incluyó la creación de servicios para gestionar operaciones comunes, como la actualización del estado de las reparaciones o la generación de reportes. Este enfoque facilitó la extensión y el mantenimiento del sistema a largo plazo.  
  
***D. DESARROLLO DE LA CAPA DE PRESENTACIÓN***

La interfaz de usuario fue creada con Windows Forms, priorizando la usabilidad y un diseño intuitivo. Se incluyeron elementos interactivos como botones, cuadros de texto y tablas para mostrar información relevante. Además, se implementaron validaciones en tiempo real para mejorar la experiencia del usuario y prevenir errores comunes.

***E. DESARROLLO DE FUNCIONALIDADES***

El sistema incluye funcionalidades clave como el registro y gestión de clientes, el seguimiento del estado de las reparaciones, la gestión de inventario de repuestos y la generación de reportes detallados. Cada una de estas funcionalidades fue diseñada para abordar necesidades específicas del negocio, mejorando la eficiencia operativa y la satisfacción del cliente.

***F. DESARROLLO DE PRUEBAS UNITARIAS Y DE INTEGRACIÓN***

Se realizaron pruebas unitarias para garantizar la funcionalidad de los componentes individuales y pruebas de integración para verificar la comunicación entre las capas. Estas pruebas incluyeron escenarios comunes y extremos para asegurar la estabilidad y confiabilidad del sistema.

V. RESULTADOS Y DISCUSIÓN

En esta sección se presentan los resultados de la estimación inicial y los resultados reales obtenidos luego de desarrollar el sistema de escritorio. Se contrastan ambos resultados para analizar la precisión y utilidad de las proyecciones iniciales en nuestro contexto específico.

* Líneas de código: 6,136
* Esfuerzo: 6 personas-meses
* Tiempo de desarrollo: 4 meses
* Número de personas: 6
* Costo:$0

***A. LÍNEAS DE CÓDIGO***

El número de líneas de código real fue de 6,136. Esto refleja el alcance y la complejidad del proyecto, alineados con las expectativas iniciales.

B. ESFUERZO  
El esfuerzo total requerido para el desarrollo fue de 6 personas-meses. Este resultado fue posible gracias al compromiso del equipo y una distribución efectiva de tareas.

C. TIEMPO DE DESARROLLO

El tiempo real de desarrollo fue de 4 meses. La planificación detallada y la organización permitieron cumplir con este plazo, garantizando una implementación eficiente.

D. COSTO TOTAL

El costo total del proyecto fue de $0, ya que el sistema fue desarrollado por estudiantes como parte de un proyecto académico. Los recursos internos y el trabajo voluntario permitieron minimizar los costos.

VI. DESAFÍOS Y SOLUCIONES

Durante el desarrollo del sistema de reparación de dispositivos móviles, enfrentamos varios desafíos que requirieron soluciones efectivas e innovadoras. Uno de los principales retos fue garantizar una integración eficiente entre las diferentes capas del sistema. Para lograrlo, definimos interfaces claras y utilizamos un enfoque modular en el diseño de la arquitectura, lo que permitió una comunicación fluida entre la capa de presentación, la lógica de negocio y la capa de datos. Además, el manejo del estado de las entidades resultó ser un desafío significativo. Se implementaron patrones para sincronizar cambios y asegurar que la información reflejada en la base de datos estuviera actualizada y libre de conflictos de concurrencia. En cuanto a la validación y manejo de errores, se estableció un sistema centralizado para identificar, registrar y manejar errores de manera efectiva, proporcionando mensajes claros al usuario y garantizando la estabilidad del sistema. Por último, la implementación de pruebas exhaustivas permitió identificar y corregir errores en etapas tempranas, asegurando la calidad del sistema entregado.

VII. LECCIONES APRENDIDAS

El desarrollo del sistema de reparación de dispositivos móviles nos dejó importantes aprendizajes. Primero, comprendimos la vital importancia de una planificación detallada. Una clara definición de los requisitos y un diseño previo facilitaron el desarrollo y ayudaron a prevenir problemas durante la implementación. Segundo, la creación de modelos, como diagramas de flujo y ERD, antes de iniciar el desarrollo del sistema, resultó crucial para identificar posibles inconvenientes de diseño y estructuración de datos. Tercero, las pruebas exhaustivas, tanto unitarias como de integración, fueron fundamentales para garantizar la estabilidad y funcionalidad del sistema. La integración de estas pruebas en el flujo de trabajo permitió identificar errores rápidamente y mantener un alto estándar de calidad. Por último, destacamos la importancia de la flexibilidad y adaptación. Durante el desarrollo, surgieron problemas inesperados que demandaron soluciones creativas y ajustes en nuestra metodología. Estas experiencias nos ayudaron a mejorar nuestras habilidades en resolución de problemas y a entender mejor los desafíos asociados al desarrollo de software, preparando al equipo para futuros proyectos más complejos.

VIII. CONCLUSIÓN  
El desarrollo del sistema de escritorio para la gestión de reparaciones de dispositivos móviles ha sido una experiencia enriquecedora y educativa para nuestro equipo. Este proyecto no solo nos permitió aplicar y consolidar conocimientos técnicos, sino que también fortaleció nuestras habilidades de trabajo en equipo y resolución de problemas. Logramos diseñar e implementar un sistema robusto y funcional que cumple con las necesidades del negocio objetivo, ofreciendo una solución segura y eficiente para la gestión de datos. Las lecciones aprendidas y los desafíos superados durante este proceso han ampliado nuestra perspectiva y nos han preparado mejor para abordar futuros proyectos en el campo del desarrollo de software.

IX. AGRADECIMIENTOS  
Queremos expresar nuestro más sincero agradecimiento a nuestro docente, cuya guía y conocimientos fueron fundamentales para el éxito de este proyecto. Sus enseñanzas y retroalimentación nos ayudaron a superar obstáculos y a crecer como desarrolladores. También agradecemos profundamente a cada uno de los miembros del equipo por su dedicación, esfuerzo y colaboración hicieron posible la culminación exitosa de este sistema.  
  
X. REFERENCIAS

1. [Base de datos sql](https://github.com/Gabo-UG/ReparacionesMovil/tree/main/Code/CapaDatos/SQLconsultas)
2. [Documentacion code Visual studio](https://github.com/Gabo-UG/ReparacionesMovil/tree/main/Code)
3. [Documentacion github](https://github.com/Gabo-UG/ReparacionesMovil/)
4. [Diagrama de componentes](https://github.com/Gabo-UG/ReparacionesMovil/tree/main/Documentacion)
5. [Diagrama de despliegue](https://github.com/Gabo-UG/ReparacionesMovil/tree/main/Documentacion)