Desarrollo y Evaluación de un Sistema de Punto de Venta para Pequeñas y Medianas Empresas Utilizando Arquitectura en Capas

A. LINDAO1, A. LOPEZ1, J. MEDINA1, J. QUINDE1, J. VILLACRES1, J. ZAMBRANO1

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

Emails: [alan.lindaoh@ug.edu.ec](mailto:%61%6ca%6e.li%6eda%6f%68@u%67%2ee%64%75%2e%65c), [anthony.lopezr@ug.edu.ec](mailto:%61%6e%74%68%6f%6e%79.%6co%70e%7a%72@%75g%2ee%64%75.%65c), [julio.medinac@ug.edu.ec](mailto:%6aul%69%6f%2e%6de%64i%6eac@%75%67%2e%65%64%75%2eec), [jeremy.quindegal@ug.edu.ec](mailto:%6a%65%72e%6d%79%2eq%75%69nd%65%67%61%6c@%75g.e%64%75%2ee%63), [joshua.villacresbot@ug.edu.ec](mailto:%6a%6fs%68%75%61%2ev%69%6cl%61%63%72es%62%6f%74@u%67.%65%64%75%2e%65c), [juan.zambranoqui@ug.edu.ec](mailto:%6a%75%61n.z%61%6dbra%6e%6fq%75i@%75%67%2eedu.e%63)

RESUMEN En el presente artículo, se describe el desarrollo de un sistema de punto de venta (POS) diseñado para pequeñas y medianas empresas, abordando los desafíos y soluciones implementadas durante el proceso. Utilizando una arquitectura en capas con Estratificación Estricta, se dividió el sistema en tres capas principales: la capa de datos, la capa de negocio y la capa de presentación. Cada capa fue diseñada para cumplir con responsabilidades específicas, mejorando la modularidad y mantenibilidad del sistema. La capa de datos empleó el patrón Repositorio, la capa de negocio utilizó el patrón EntityState, y la capa de presentación adoptó la arquitectura Modelo-Vista-Modelo de Vista (MVVM). Para el desarrollo, se utilizaron tecnologías como C# 6.0, SQL Server 2022, y Visual Studio, entre otras. El proyecto enfrentó desafíos relacionados con la integración entre capas, el manejo del estado de entidades y el rendimiento de consultas, los cuales fueron superados mediante soluciones efectivas y pruebas exhaustivas. Se realizaron pruebas unitarias, de integración para asegurar la calidad del sistema. El proceso proporcionó valiosas lecciones sobre la importancia de la planificación, el modelado previo, la flexibilidad y el manejo de errores. Los resultados y la experiencia obtenida enriquecieron nuestras habilidades en desarrollo de software.

PALABRAS CLAVES Arquitectura en Capas, C# 6.0, Desarrollo de Software, EntityState, Estratificación Estricta, MVVM, Optimización de Consultas, Patrón Repositorio, Pruebas de Integración, Pruebas Unitarias, SQL Server, Visual Studio

I. INTRODUCCIÓN

En la actualidad, las empresas independientemente de su tamaño enfrentan el desafío de gestionar eficientemente la gran cantidad de información generada por sus usuarios. La solución más común es implementar sistemas personalizados que les permitan manejar sus operaciones de manera efectiva. La tecnología juega un papel significativo para satisfacer las exigentes demandas de la sociedad. Esta tecnología ha influenciado enormemente en nuestras vidas, ya que todo se realiza a través de un computador. La economía gira alrededor de un mundo cibernético, donde todas las compras y transacciones bancarias se pueden realizar por medio de un computador desde la sala de su hogar. También en el aspecto político, ya que todos los servicios básicos son realizados desde un computador [[2]](#Referencia2).

Como futuros ingenieros de software, tenemos la responsabilidad de comprender las necesidades de nuestros clientes y desarrollar sistemas que las satisfagan. En el marco de la asignatura de Construcción de Software, decidimos abordar este desafío desarrollando un sistema de punto de venta (POS) destinado a pequeñas y medianas empresas.

Nuestro objetivo principal es crear una solución sencilla y eficiente para la gestión de ventas, inventario y clientes. Este proyecto no solo busca proporcionar una herramienta útil para las empresas, sino también ofrecernos la oportunidad de adquirir experiencia práctica en el desarrollo colaborativo de software.

II. TRABAJOS RELACIONADOS

Para el desarrollo de nuestro sistema de punto de venta (POS), hemos revisado varios trabajos que han abordado la creación de sistemas similares. Por ejemplo, en [[1]](#Referencia1), el autor desarrolla un sistema de punto de venta implementando la arquitectura cliente-servidor para una pequeña empresa. Además, diseñaron diagramas de clases, de secuencia y de estado, lo cual facilitó el desarrollo del sistema. Estos diagramas sirvieron como referencia para el diseño de nuestro propio sistema. También consideramos el trabajo de [[2]](#Referencia2), en el cual se desarrolla un sistema web, permitiendo que sea accesible desde cualquier lugar con una conexión a internet. Sin embargo, decidimos que nuestro sistema sea una aplicación de escritorio con un servidor de base de datos local, lo cual nos permite un mayor control sobre los datos y la seguridad del sistema.

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. ESTIMACIÓN DE COSTOS UTILIZANDO COCOMO I

Para realizar una estimación de costos utilizando el modelo COCOMO básico, consideramos las siguientes características de nuestro proyecto:

* Tamaño del Proyecto: 8,000 líneas de código (8 KLOC)
* Tipo de Proyecto: Orgánico

1) CÁLCULO DE ESFUERZO (PERSONA-MESES)

La fórmula para calcular el esfuerzo en un proyecto Orgánico es:

Aplicamos los datos:

Realicemos el cálculo:

2) CÁLCULO DE TIEMPO DE DESARROLLO (MESES)

La fórmula para calcular el tiempo de desarrollo es:

Aplicamos el esfuerzo calculado:

Realicemos el cálculo:

3) NÚMERO DE PERSONAS REQUERIDAS

La fórmula para calcular el número de personas requeridas es:

Aplicamos los valores calculados:

4) CÁLCULO DE COSTOS

Para calcular los costos, consideramos una tarifa promedio de $500 por persona-mes.

5) RESUMEN DE LA ESTIMACIÓN

* Esfuerzo Estimado: 20.14 persona-meses
* Tiempo de Desarrollo Estimado: 8.35 meses
* Número de Personas Requeridas Estimado: 2.41 personas
* Costo Total Estimado: $10,070

B. HERRAMIENTAS Y TECNOLOGÍA

1) LENGUAJES DE PROGRAMACIÓN

Para el desarrollo de nuestro sistema, seleccionamos C# 6.0 [[3]](#c3) como el lenguaje de programación principal. La elección de C# se basó en su robustez y amplia adopción en el desarrollo de aplicaciones empresariales.

2) BASE DE DATOS

Utilizamos SQL Server 2022 [[4]](#sql4) para la gestión de la base de datos. Esta versión nos proporcionó características avanzadas de seguridad y rendimiento.

3) ENTORNO DE DESARROLLO

Para el desarrollo del software, empleamos las siguientes tecnologías:

* Visual Studio: Es un entorno de desarrollo integrado (IDE) [[5],](#visualEstudio5) el cual nos proporcionó una plataforma robusta y rica en características para escribir, depurar y probar nuestro código.
* GitHub: Es un repositorio de código [[6]](#github6), este nos facilitó la colaboración entre los miembros del equipo.

4) PAQUETES Y LIBRERÍAS

Para desarrollar nuestro proyecto hicimos uso de varios paquetes y librerías:

* SqlClient: Proveedor de datos de .NET para SQL Server [[7]](#client7), que nos permitió conectar nuestra aplicación con la base de datos y realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) de manera eficiente.
* NUnit: El cual usamos como nuestro framework de pruebas unitarias para .NET [[8]](#nunit8). NUnit facilitó la creación y ejecución de pruebas automatizadas para asegurar la calidad del código.
* NUnit3TestAdapter: El cual se integró con la ventana de pruebas unitarias de Visual Studio [[9]](#testadapter9), permitiéndonos ejecutar y visualizar los resultados de nuestras pruebas unitarias directamente en el IDE.
* Microsoft.NET.Test.Sdk: Proporcionó la infraestructura necesaria para ejecutar pruebas unitarias en proyectos .NET dentro de Visual Studio [[10]](#sdk10).

C. DIAGRAMA DE COMPONENTES

El diagrama de componentes en la figura [1](#imagen1) ilustra las principales partes del sistema de punto de venta y cómo interactúan entre sí.

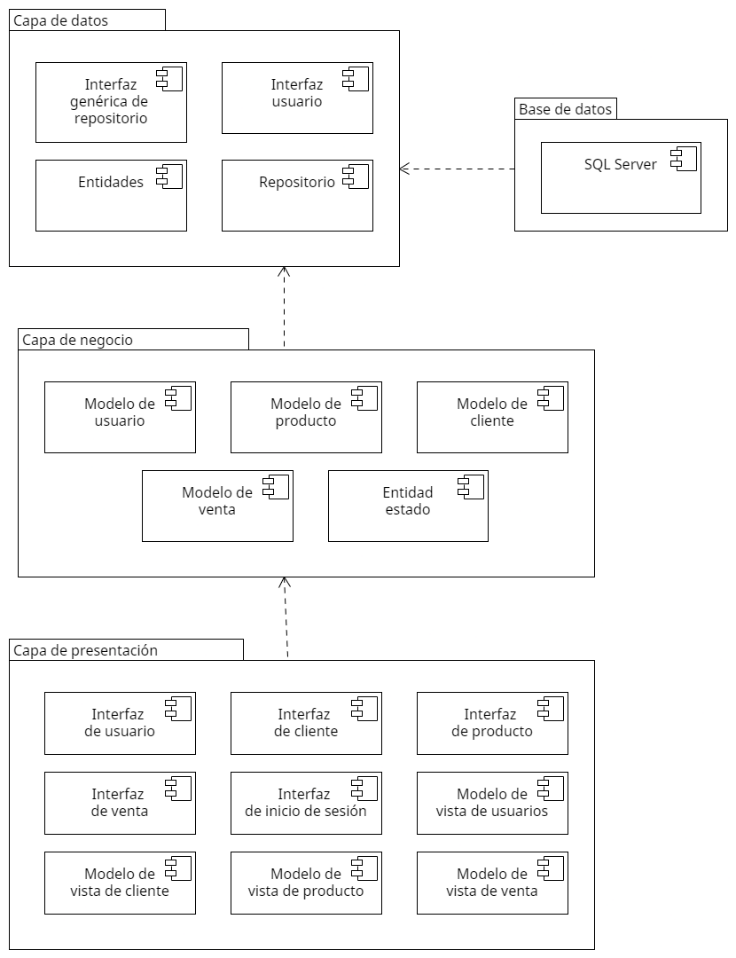


FIGURA 1.  Diagrama de componentes del sistema.

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

1) CAPA DE PRESENTACIÓN

La capa de presentación es responsable de la interacción con el usuario. Define las interfaces para usuarios, clientes, productos y ventas, así como para el inicio de sesión. Los modelos de vista en esta capa manejan la lógica de presentación y la interacción de la interfaz de usuario, proporcionando una separación clara entre la lógica de presentación y la lógica de negocio.

* Interfaz de Usuario: Define la interfaz para que los usuarios interactúen con el sistema.
* Interfaz de Cliente: Define la interfaz para que los clientes interactúen con el sistema.
* Interfaz de Producto: Define la interfaz para la interacción con los productos.
* Interfaz de Venta: Define la interfaz para la interacción con las ventas.
* Interfaz de Inicio de Sesión: Gestiona la lógica y la interfaz para el inicio de sesión de los usuarios.
* Modelo de Vista de Usuarios: Implementa la lógica de presentación y la interacción de la vista de usuarios.
* Modelo de Vista de Cliente: Implementa la lógica de presentación y la interacción de la vista de clientes.
* Modelo de Vista de Producto: Implementa la lógica de presentación y la interacción de la vista de productos.
* Modelo de Vista de Venta: Implementa la lógica de presentación y la interacción de la vista de ventas.

2) CAPA DE NEGOCIO

La capa de negocio contiene la lógica de negocio del sistema. Aquí se definen los modelos que manejan la lógica específica de cada entidad, como usuarios, productos, clientes y ventas. También se encuentra la entidad estado, que facilita el manejo del estado de las entidades dentro del contexto de la aplicación.

* Modelo de Producto: Gestiona la lógica de negocio relacionada con los productos.
* Modelo de Cliente: Maneja la lógica de negocio relacionada con los clientes.
* Modelo de Venta: Gestiona la lógica de negocio relacionada con las ventas.
* Entidad Estado: Permite rastrear y manejar el estado de las entidades en el contexto de la aplicación.

3) CAPA DE DATOS

La capa de datos se encarga de la gestión y acceso a los datos del sistema. Incluye las entidades que representan las tablas de la base de datos y el repositorio que contiene la lógica para interactuar con la base de datos a través de interfaces específicas y genéricas. Esta capa se comunica directamente con la base de datos para realizar operaciones de almacenamiento, recuperación, actualización y eliminación de datos.

* Interfaz Genérica de Repositorio: Define métodos generales para interactuar con la base de datos sin necesidad de conocer la implementación específica de los datos.
* Interfaz Usuario: Define los métodos específicos para interactuar con los datos de los usuarios.
* Entidades: Representa las estructuras de datos que corresponden a las tablas de la base de datos.
* Repositorio: Implementa la lógica para acceder y manipular los datos en la base de datos.

4) BASE DE DATOS

La base de datos almacena la información del usuario, productos, ventas y clientes. Este componente interactúa con otros componentes para recuperar y guardar datos.

* SQL Server: Sistema de gestión de bases de datos relacional donde se almacenan los datos del sistema.

D. DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue en la figura [2](#imagen2) ilustra la arquitectura física del sistema de punto de venta (POS). Este diagrama es esencial para comprender la estructura del sistema y garantizar que todos los componentes funcionen de manera coherente y eficiente.

A computer server diagram with text

Description automatically generated with medium confidence

FIGURA 2.  Diagrama de despliegue.

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

1) PC

Es el componente principal donde se ejecuta el sistema de punto de venta. Es el punto central de operaciones, desde donde el usuario interactúa con el sistema para llevar a cabo todas las transacciones y gestiones del negocio. El PC tiene el siguiente componente de software:

* Sistema de Punto de Venta: El sistema de punto de venta es la aplicación de software instalada en el PC. Está diseñada para gestionar las operaciones diarias de ventas, inventarios, y clientes. Dentro del sistema de punto de venta, se pueden identificar tres capas fundamentales:
  + Capa de Presentación: Es la capa que interactúa directamente con el usuario. Incluye todas las interfaces de usuario que permiten la interacción con el sistema, como ventanas de ventas, formularios de entrada de datos, y menús.
  + Capa de Negocio: Esta capa maneja la lógica del negocio. Procesa las solicitudes provenientes de la capa de presentación y se comunica con la capa de datos para realizar operaciones de lectura y escritura.
  + Capa de Datos: Esta capa es responsable de la gestión de datos del sistema. Incluye componentes que interactúan con la base de datos para realizar operaciones CRUD.

2) servidor local de base de datos

El servidor local de base de datos se ejecuta en la misma máquina que el sistema de punto de venta, pero está separado lógicamente para mejorar la organización y gestión de datos.

* SQL Server: Es el motor de base de datos [[4]](#sql4) utilizado para almacenar toda la información necesaria para el funcionamiento del sistema de punto de venta. Esto incluye datos de productos, ventas, clientes, y cualquier otra información relevante para el negocio.

IV. DESARROLLO

***A. DIVISIÓN DEL TRABAJO***

Nos organizamos en tres equipos: capa de presentación, capa de negocio y capa de datos. Cada equipo tenía sus tareas específicas, pero manteníamos reuniones semanales para coordinar el progreso y resolver problemas con los que nos íbamos topando.

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

Implementamos el patrón Repositorio para gestionar el acceso a la base de datos. Este patrón proporcionó una interfaz abstracta para interactuar con los datos, permitiendo operaciones CRUD (Crear, Leer, Actualizar y Eliminar) de manera eficiente. La capa de datos se encargó de las consultas a la base de datos, la ejecución de comandos y la manipulación de datos, garantizando que la lógica de acceso a datos estuviera desacoplada de la lógica de negocio.

***C. DESARROLLO DE LA CAPA DE NEGOCIO***

En esta capa, utilizamos el patrón EntityState para gestionar el estado de las entidades a lo largo de las operaciones del sistema. EntityState permitió el seguimiento y la sincronización de los cambios en las entidades, asegurando que las actualizaciones se reflejasen correctamente en la base de datos y facilitando la implementación de las reglas de negocio.

***D. DESARROLLO DE LA CAPA DE PRESENTACIÓN***

Aplicamos la arquitectura Modelo-Vista-Modelo de Vista (MVVM) para separar la lógica de la interfaz de usuario de la lógica de presentación. La capa de presentación se encargó de la interacción con el usuario, presentando la información de manera clara y gestionando las solicitudes de entrada. MVVM permitió una gestión más limpia y modular de la interfaz de usuario, facilitando el desarrollo y mantenimiento de las vistas y sus correspondientes modelos de vista.

***E. DESARROLLO DE FUNCIONALIDADES***

Durante el desarrollo, nos centramos en implementar las funcionalidades clave del sistema, asegurando que cada componente funcionara de acuerdo con los requisitos definidos:

* Gestión de Productos: Implementamos funcionalidades para agregar, editar, eliminar y visualizar productos. Esto incluyó la creación de interfaces para la entrada de datos y la integración con la capa de datos para almacenar y recuperar información.
* Gestión de Usuarios y Clientes: Desarrollamos módulos para manejar la información de usuarios y clientes, incluyendo la autenticación de usuarios y la gestión de perfiles de cliente.
* Procesos de Venta: Implementamos las funcionalidades necesarias para registrar y gestionar ventas.

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

Realizamos pruebas para verificar la funcionalidad de cada componente del sistema. Las pruebas se integraron en el flujo de trabajo de desarrollo para asegurar que cada nuevo cambio no introdujera errores en el sistema.

* Pruebas unitarias: Utilizamos NUnit [[8]](#nunit8) para realizar pruebas unitarias en los componentes individuales del sistema. Estas pruebas verificaron la correcta implementación de funciones y métodos en cada capa. Las pruebas unitarias nos permitieron identificar y corregir errores, asegurando que cada componente funcionara según lo esperado.
* Pruebas de integración: Realizamos pruebas de integración para validar la interacción entre las distintas capas del sistema. Estas pruebas confirmaron que los datos se transmitieran correctamente entre la capa de datos y la base de datos.

V. RESULTADOS Y DISCUSIÓN

En esta sección se presentan los resultados de la estimación de costos utilizando el modelo COCOMO I y los resultados reales obtenidos luego de desarrollar el sistema de punto de venta. Se contrastan ambos resultados para analizar la precisión y utilidad del modelo en nuestro contexto específico.

* Resultados de laestimación con COCOMO I
  + Líneas de código estimadas: 8000
  + Esfuerzo Estimado: 20.14 persona-meses
  + Tiempo de Desarrollo Estimado: 8.35 meses
  + Número de Personas Requeridas Estimado: 2.41 personas
  + Costo Total Estimado: $10,070
* Resultados reales del proyecto
  + Líneas de código: 9,313
  + Esfuerzo: 0.67 persona-meses
  + Tiempo de desarrollo: 4 meses
  + Número de personas: 6 personas
  + Costo total: $0

A. LÍNEAS DE CÓDIGO

El número de líneas de código estimadas fue de 8000, mientras que el proyecto real terminó con 9,313 líneas de código. Esto indica que el alcance del proyecto fue ligeramente mayor de lo previsto inicialmente.

B. ESFUERZO

La estimación inicial de esfuerzo utilizando COCOMO I fue significativamente mayor (20.14 persona-meses) en comparación con el esfuerzo real requerido (0.67 persona-meses). Esta gran discrepancia sugiere que el modelo COCOMO I puede sobreestimar el esfuerzo necesario para proyectos de tamaño y complejidad similares al nuestro.

C. TIEMPO DE DESARROLLO

La estimación de tiempo de desarrollo según COCOMO I fue de 8.35 meses, mientras que el tiempo real de desarrollo fue de 4 meses. La diferencia significativa en el tiempo de desarrollo puede atribuirse a la experiencia y la habilidad del equipo de desarrollo, así como a una planificación y ejecución eficientes del proyecto.

D. NÚMERO DE PERSONAS REQUERIDAS

COCOMO I estimó que se necesitarían aproximadamente 3.41 personas para completar el proyecto. En la práctica, 6 personas trabajaron en el proyecto. Esta diferencia puede reflejar una distribución del trabajo más colaborativa y una asignación de tareas que permitió un desarrollo más rápido y eficiente.

E. COSTO TOTAL

El costo total estimado por COCOMO I fue de $10,070, mientras que el costo real fue de $0. Esta diferencia se debe a que el sistema fue desarrollado por estudiantes como proyecto final de la materia de construcción de software. Los desarrolladores trabajaron de manera voluntaria y se utilizaron recursos internos sin costo adicional, factores que no fueron contemplados en el modelo COCOMO I.

VI. DESAFÍOS Y SOLUCIONES

Durante el desarrollo del sistema, enfrentamos varios desafíos que requirieron soluciones innovadoras para garantizar el éxito del proyecto. A continuación, se describen algunos de los principales problemas y cómo los abordamos.

A. INTEGRACIÓN ENTRE CAPAS

El desafío era asegurar que las diferentes capas del sistema interactuaran correctamente sin violar los principios de separación de responsabilidades, para lo cual implementamos interfaces bien definidas entre la capa de datos, la capa de negocio y la capa de presentación. Ajustamos las interfaces según fuera necesario para resolver problemas de integración.

B. MANEJO DEL ESTADO DE ENTIDADES

Debíamos sincronizar correctamente el estado de las entidades entre la capa de negocio y la base de datos, especialmente con cambios concurrentes. Adoptamos el patrón EntityState para gestionar el seguimiento y actualización del estado de las entidades. Implementamos mecanismos para manejar conflictos de concurrencia y aseguramos la coherencia de los datos mediante transacciones adecuadas.

C. VALIDACIÓN Y MANEJO DE ERRORES

Con esto buscábamos garantizar que el sistema manejara errores de manera robusta y proporcionara mensajes de validación claros al usuario. Implementamos un manejo de errores centralizado y validaciones en la capa de presentación. Desarrollamos pruebas exhaustivas para validar los casos de borde y asegurar que los errores fueran gestionados de manera adecuada.

VII. LECCIONES APRENDIDAS

A lo largo del desarrollo del sistema, aprendimos varias lecciones valiosas:

A. IMPORTANCIA DE LA PLANIFICACIÓN

La planificación detallada y la definición clara de requisitos son cruciales para el éxito del proyecto. Un diseño bien planificado facilita la implementación y reduce el riesgo de problemas durante el desarrollo.

B. BENEFICIOS DEL MODELADO PREVIO

La creación de diagramas de flujo y diagramas ERD antes de comenzar la codificación ayudó a visualizar y resolver problemas de diseño temprano. El modelado previo es una práctica recomendada para evitar problemas en etapas posteriores.

C. NECESIDAD DE PRUEBAS EXHAUSTIVAS

Las pruebas exhaustivas, incluyendo pruebas unitarias y de integración, son esenciales para garantizar la calidad y fiabilidad del sistema. La integración de pruebas en el flujo de trabajo de desarrollo ayuda a identificar y corregir errores de manera oportuna.

D. FLEXIBILIDAD Y ADAPTACIÓN

La capacidad de adaptarse a cambios y resolver problemas imprevistos es fundamental. Aprendimos a ajustar nuestro enfoque y a buscar soluciones creativas para superar los desafíos que surgieron durante el desarrollo.

VIII. CONCLUSIÓN

El desarrollo del sistema de punto de venta ha sido una experiencia educativa valiosa que nos ha permitido aplicar y profundizar nuestros conocimientos en arquitectura de software, desarrollo de bases de datos y pruebas de software. La implementación de una arquitectura en capas, junto con el uso de herramientas y tecnologías avanzadas, ha resultado en un sistema robusto y eficiente. Las lecciones aprendidas y los desafíos superados durante el proyecto han enriquecido nuestra experiencia y nos han preparado mejor para futuros proyectos en el campo del desarrollo de software.

IX. AGRADECIMIENTOS

Queremos expresar nuestro agradecimiento a nuestro profesor por sus valiosas enseñanzas que fueron fundamentales para afrontar con éxito el desarrollo del proyecto.

Agradecemos también a nuestros compañeros de equipo por su dedicación, esfuerzo y colaboración durante todo el proceso. Su compromiso y trabajo en equipo fueron clave para la realización del sistema, y el ambiente de cooperación y apoyo mutuo contribuyó significativamente al éxito del proyecto.

X. REFERENCIAS

1. M. R. Uzho. (2011). “Sistema para Puntos de Venta en Micro Empresas”. Escuela Politécnica Nacional, Escuela de Formación de Tecnólogos. Disponible en: <http://bibdigital.epn.edu.ec/handle/15000/4395>
2. C. J. Aguilar. (2011). “Sistema de Punto de Venta para Una Mercería”. Instituto Tecnológico de Tuxtla Gutiérrez. Disponible en: <http://repositoriodigital.tuxtla.tecnm.mx/xmlui/handle/123456789/315>

1. [Documentación oficial de C#](https://docs.microsoft.com/en-us/dotnet/csharp/" \t "_new)

1. [Documentación oficial de SQL Server](https://docs.microsoft.com/en-us/sql/sql-server/sql-server-2022" \t "_new)

1. [Documentación oficial de Visual Studio](https://docs.microsoft.com/en-us/visualstudio/" \t "_new)

1. [Documentación oficial de GitHub](https://docs.github.com/en" \t "_new)

1. [Documentación de SqlClient](https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient" \t "_new)

1. [Documentación oficial de NUnit](https://nunit.org/" \t "_new)

1. [Documentación de NUnit3TestAdapter](https://github.com/nunit/nunit3-vs-adapter" \t "_new)

1. [Documentación de Microsoft.NET.Test.Sdk](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-nunit" \t "_new)