TÍTULO

DISEÑO E IMPLEMENTACIÓN BASE DE DATOS PARA EL CONTROL DE PIEZAS Y COMPONENTES DE PRUEBA PARA AERONAVES

AUTOR

Luis Gabriel Forte Forte

Esta edición electrónica ha sido realizada en 2025

Directores	Dr. Pedro Javier Zarco Periñán; Rafael Márquez Jiménez
Instituciones	Universidad Internacional de Andalucía; Universidad de Granada; Universidad de Málaga; Universidad de Almería
Curso	*Máster Universitario en Transformación Digital de Empresas (2023/24)*
©	Luis Gabriel Forte Forte
©	De esta edición: Universidad Internacional de Andalucía
Fecha documento	2024

Máster en Transformación digital de empresas

TRABAJO FIN DE MÁSTER

Universidad Internacional de Andalucía
Centro: Oficina de Estudios de Posgrado.

"DISEÑO E IMPLEMENTACION BASE DE DATOS PARA EL CONTROL DE PIEZAS Y COMPONENTES DE PRUEBA PARA AERONAVES"

Itinerario: Sector Energético

Curso: 2023/2024

Modalidad: Trabajo Técnico

Alumno/a: Luis Gabriel Forte Forte

Director/es:

- Dr. D. Pedro Javier Zarco Periñán
- D. Rafael Márquez Jimenez

SIGLAS Y ACRÓNIMOS

Abreviatura	Significado Original	Significado Español
MTDE	Máster Transformación Digital de Empresas	
SQL	Structured Query Language	Lenguaje de Consulta Estructurada
SGBD	Sistema Gestión Base de Datos	
DMS	Document Management System	Sistema de Gest
PN	Part Number	Número de parte
S/N	Serial Number	Número de serie
TFM	Trabajo Fin de Máster	
CRUD	Create,Read,Update,Delete	Crear, Leer, Actualizar, Eliminar
UAS	Unmanned Aircraft System	Sistema de Aeronave no Tripulada
ERP	Enterprise Resource Planning	Planificación de Recursos Empresariales
DGV	DataGridView	Vista de cuadrícula de datos
LDAP	Lightweight Directory Access Protocol	Protocolo ligero de acceso a directorios
HTTPS	Hypertext Transfer Protocol Secure	protocolo de transferencia de hipertexto seguro
AES	Advanced Encryption Standard	Algoritmo de Cifrado Simétrico
ReqView	Requirements View	Vista de Requisitos

Índice de Figuras

Índice de tablas

AGRADECIMIENTOS

Quisiera expresar mi más sincero agradecimiento a todas las personas y entidades que han hecho posible la realización de este proyecto. A mi familia, en especial a mi esposa e hijos, quienes me han brindado su apoyo y ánimo en los momentos más difíciles, permitiéndome llegar a la conclusión de este camino.

A la Universidad Internacional de Andalucía, por la organización de este Máster tan actualizado sobre la Transformación Digital de Empresas. Agradezco a todos los profesores, cuyas materias, aunque diversas, resultaron sumamente interesantes y guardan una coherencia que refleja la actualidad de los temas tratados. Una mención especial para Pedro Zarco, tutor de este trabajo, por su invaluable ayuda y guía a lo largo de todo el proceso de desarrollo.

A la empresa AERTEC, por acogerme durante mi estancia dual y permitirme aprender sobre las peculiaridades del sector aeronáutico. Mi agradecimiento también a todos los empleados, en particular a Jesús Gutiérrez, quien, como tutor en la empresa, me enseñó aspectos clave de la programación y el funcionamiento de las bases de datos.

INDICE

1.MEMORIA DESCRIPTIVA DEL PROYECTO.

El presente Trabajo de Fin de Máster (TFM) describe en detalle el diseño e implementación de una base de datos destinada a unificar el repositorio de ofertas y proyectos, así como a inventariar los números de serie de los productos fabricados, lo cual es fundamental para gestionar futuros encargos de los clientes. Este sistema tiene como objetivo principal la automatización y gestión eficiente de los procesos productivos de la empresa.

Para llevar a cabo este proyecto, se ha utilizado MariaDB como sistema de gestión de bases de datos debido a su robustez, escalabilidad y eficiencia. El desarrollo del software se ha realizado utilizando el lenguaje de programación C#, que ofrece una integración sólida con MariaDB y proporciona las herramientas necesarias para crear aplicaciones robustas y eficientes.

La base de datos diseñada no solo centraliza la información de ofertas y proyectos, sino que también asegura una trazabilidad completa de los productos desde su fabricación hasta la entrega al cliente. Esta trazabilidad es crucial para cumplir con los requerimientos técnicos solicitados por los clientes en las ofertas y asegurar que cada producto cumple con los estándares de calidad establecidos.

Uno de los aspectos más destacados de este proyecto es la capacidad de inventariar los números de serie de cada producto fabricado. Esta funcionalidad permite a la empresa llevar un control detallado y preciso de cada producto, facilitando su seguimiento y gestión en futuros encargos. Además, esta característica mejora significativamente la capacidad de respuesta de la empresa ante consultas y reclamaciones de los clientes, asegurando un servicio postventa de alta calidad.

En resumen, este TFM presenta una solución integral para la gestión de ofertas, proyectos y productos fabricados, utilizando MariaDB y C#. La implementación de esta base de datos no solo mejora la eficiencia y la automatización de los procesos productivos, sino que también asegura el cumplimiento de los requerimientos técnicos y la trazabilidad completa de los productos. Este proyecto representa un paso significativo hacia la digitalización y modernización de la gestión empresarial.

1.1 CONTEXTO AERTEC

AERTEC es una empresa con una amplia experiencia y reconocimiento en la industria aeroespacial, especializada en el ciclo industrial del avión, que abarca desde el diseño conceptual hasta la instalación final.

Como proveedor destacado en áreas clave como alas y cabina, AERTEC ofrece soluciones integrales que cubren todo el ciclo de vida del diseño del avión. Esto incluye desde la personalización de cabinas, como las del A350, hasta el desarrollo de sistemas complejos para modelos como el A380.

La empresa se especializa en el diseño eléctrico, ofreciendo un rango completo de servicios desde el análisis y definición funcional hasta la instalación y pruebas de sistemas eléctricos. Entre sus capacidades se encuentran el diseño de mazos eléctricos, la validación y verificación de estos sistemas, y la realización de pruebas de seguridad. AERTEC también se dedica a la ingeniería de sistemas de plataformas, incluyendo el soporte en sistemas de

control de vuelo, sistemas electrónicos, de actuación y de misión, además de ofrecer soluciones para la validación y verificación del desarrollo de software aeronáutico.

Una de las innovaciones destacadas de AERTEC es su sistema de producción industrial completamente operativo e implementado, que incluye la producción de mazos de cables eléctricos. La empresa ha desarrollado "i-Tabla", una herramienta que optimiza la fabricación de cables al proporcionar un acceso digital e interactivo a órdenes de producción, mejorando así la eficiencia y reduciendo costos y tiempos de desarrollo.

Además, AERTEC ha creado un equipo que permite la monitorización y validación de tensiones y frecuencias en mazos eléctricos durante la fase de cableado de aviones, lo cual es crucial para las tareas de mantenimiento y reparación. Este equipo destaca por su capacidad de operar de manera segura con alimentación interna y registrar datos de forma autónoma gracias a su conectividad WiFi y tarjeta SD.

En el ámbito de los sistemas aéreos no tripulados (UAS), AERTEC se especializa en el diseño, desarrollo y operación de sistemas de ala fija, con una alta eficiencia aerodinámica para misiones de largo alcance. La empresa se posiciona como una autoridad en el diseño de UAS, con una sólida experiencia que le permite ofrecer soluciones innovadoras y personalizadas que cumplen con los más altos estándares del mercado. Su participación en proyectos nacionales e internacionales de UAS le proporciona una visión avanzada de las tendencias tecnológicas actuales y futuras.

1.2 OBJETIVOS

El objetivo de este TFM es diseñar e implementar una base de datos utilizando MariaDB, una solución de código abierto. Actualmente, la gestión de ofertas y proyectos en AERTEC se lleva a cabo mediante un archivo Excel, que contiene aproximadamente 15 columnas y unas 600 filas. Sin embargo, este método presenta limitaciones en la gestión de la información relacionada con las piezas y componentes de fabricación, es preciso mayor control sobre los mismos, así como en la trazabilidad y la documentación asociada. Es por lo que se va a seguir las siguientes fases

- Analizar el proceso actual de gestión de ofertas y proyectos en AERTEC, identificando las limitaciones y deficiencias del sistema de seguimiento basado en Excel.
- Diseñar una estructura de base de datos en MariaDB que permita almacenar y gestionar de manera eficiente la información relacionada con las ofertas y proyectos, incluyendo los números de serie, sus especificaciones de fabricación y certificación, así como la documentación asociada. Se incluye formularios realizados en Visual Studio para interactuar con la base de datos, diseñando prototipo para validación por parte del usuario.
- Desarrollo e implementación: En esta fase se procede a la implementación de la base de datos utilizando el sistema gestor de bases de datos (SGBD) seleccionado. Se escriben los scripts SQL (Structured Query Language), necesarios para crear las tablas, definir las restricciones y los índices, así como para cargar los datos iniciales.
- Configuración de base de datos en equipo local para obtener una versión funcional sin atender aún a las limitaciones de conectividad de red de la empresa.
- Pruebas y validación: Se realizan pruebas exhaustivas para validar el funcionamiento y la integridad de la base de datos. Se verifica que cumpla con los requisitos funcionales y no

funcionales establecidos previamente. Se corrigen los errores o deficiencias identificados durante esta etapa.
- Revisión funcional de parte de los usuarios finales para confirmar la usabilidad de la aplicación.
- Integración con otros sistemas.
- Instalación y configuración del sistema de base de datos en servidor corporativo, accesible desde dispositivos de la red interna.

1.3 ORIGEN DE LAS BASES DE DATOS RELACIONALES Y SUS GESTORES

El concepto de bases de datos relacionales fue propuesto por Edgar F. Codd en 1969, quien, trabajando para IBM, publicó su influyente artículo titulado (Codd, 1969)"A Relational Model of Data for Large Shared Data Banks" en 1970 [1]. Este artículo introdujo el modelo relacional como una forma novedosa y más eficiente de gestionar grandes cantidades de datos compartidos. Antes del modelo relacional, los datos se almacenaban principalmente en modelos jerárquicos o de red, que presentaban dificultades en cuanto a flexibilidad, redundancia y complejidad.

El modelo relacional de Codd se basa en la teoría de conjuntos y en la lógica de predicados, lo que permite una organización más natural y matemática de los datos. La propuesta de Codd no solo revolucionó la manera en que los datos podían almacenarse y manipularse, sino que también proporcionó una base sólida para la creación de lenguajes de consulta de alto nivel, como SQL. Este modelo sentó las bases para la mayoría de los (SGBD) modernos, que permiten a los usuarios interactuar con los datos de manera eficiente y lógica.

1.3.1 Definición de Bases de Datos Relacionales

Una base de datos relacional es una colección organizada de datos estructurados en tablas, donde las relaciones entre los datos se representan mediante claves primarias y foráneas. Cada tabla, también llamada relación, consta de filas y columnas. Las columnas representan los atributos o campos de la tabla, mientras que las filas representan las tuplas o registros [2].

El modelo relacional se basa en los siguientes principios fundamentales:

- Estructura natural de los datos: Los datos se organizan en su forma más básica y directa, respetando su estructura natural. Esto facilita la consulta y la manipulación de la información.
- Teoría de conjuntos: El modelo relacional trata las tablas como conjuntos, lo que significa que no pueden existir elementos duplicados en una relación. Las operaciones en las tablas se basan en las operaciones de la teoría de conjuntos, como la unión, intersección y diferencia.
- Lógica de predicados: El modelo relacional permite la creación de consultas a través de expresiones lógicas que se evalúan para devolver resultados basados en criterios de verdad. Esto es la base del lenguaje SQL.

El modelo relacional aborda varios problemas que eran comunes en los enfoques anteriores, como la redundancia y la inconsistencia lógica. Al eliminar datos duplicados y definir reglas claras para la integridad referencial, las bases de datos relacionales aseguran que los datos permanezcan consistentes a lo largo del tiempo, sin necesidad de múltiples verificaciones manuales.

1.3.2 Datos y Metadatos en las Bases de Datos Relacionales

En el contexto de las bases de datos relacionales, es crucial diferenciar entre datos y metadatos, ya que ambos desempeñan roles fundamentales en la estructura y gestión de la información.

Un dato es una representación simbólica de un hecho o atributo que describe una entidad en particular. Los datos pueden ser de naturaleza cuantitativa (números, cantidades) o cualitativa (nombres, descripciones). En una base de datos relacional, los datos se almacenan en tablas, donde cada fila (o tupla) representa un registro, y cada columna (o atributo) representa una característica específica del registro.

Por ejemplo, en una tabla de productos, un dato podría ser el nombre del producto, el código de identificación, el precio o la cantidad disponible en inventario. Estos datos, tomados individualmente, pueden no tener mucho valor, pero cuando se agrupan en una tabla, permiten representar de manera estructurada una entidad del mundo real, como un producto en un sistema de inventario.

El valor de los datos radica en su organización y estructura. Un conjunto de datos aislados no constituye información útil hasta que se organiza y se presenta en un contexto que permita interpretarlos. En una base de datos relacional, los datos se organizan en un formato tabular, permitiendo que se realicen consultas para extraer información significativa y apoyar la toma de decisiones.

Los metadatos son datos que describen otros datos. En una base de datos relacional, los metadatos proporcionan información sobre la estructura, el esquema y las características de los datos almacenados. Mientras que los datos representan los valores específicos almacenados en las tablas, los metadatos definen cómo se organizan y estructuran esos datos dentro de la base de datos.

Por ejemplo, los metadatos en una base de datos relacional incluyen (Figura 1.1):

- Nombres de tablas y columnas: Información sobre cómo se llaman las tablas y los atributos que contienen.
- Tipos de datos: Descripción de los tipos de datos permitidos para cada columna (por ejemplo, entero, cadena de texto, fecha).
- Restricciones: Reglas que se aplican a los datos, como claves primarias, claves foráneas, restricciones de unicidad y valores nulos.
- Índices: Información sobre los índices que se utilizan para acelerar la consulta de datos.
- Relaciones entre tablas: Definición de las relaciones que existen entre las distintas tablas, como las relaciones uno a uno, uno a muchos o muchos a muchos.

En resumen, los metadatos actúan como una guía que permite a los sistemas gestores de bases de datos interpretar y manejar correctamente los datos. Estos son esenciales para la administración y operación de la base de datos, ya que permiten al sistema entender la estructura y las relaciones entre los datos.

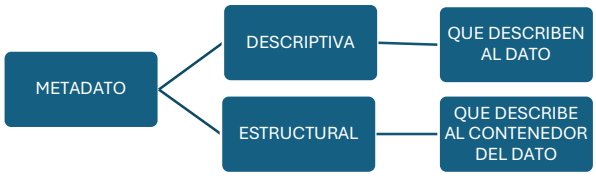


Figura 1.1 *Tipos de metadatos elaboración propia*

Los metadatos tienen un papel fundamental en la gestión eficiente de las bases de datos relacionales, ya que permiten:

- Consultas optimizadas: Los sistemas gestores de bases de datos utilizan los metadatos para optimizar la ejecución de consultas, seleccionando los índices más adecuados y aplicando las restricciones de manera eficiente.
- Integridad de los datos: Los metadatos definen las reglas que deben cumplirse para garantizar la integridad de los datos, como las claves primarias y foráneas, evitando la entrada de datos duplicados o inconsistentes.
- Mantenimiento del esquema: Los metadatos facilitan el mantenimiento y la evolución del esquema de la base de datos, permitiendo la introducción de nuevas tablas, columnas o restricciones sin afectar negativamente a los datos existentes.
- Seguridad: Los sistemas gestores de bases de datos utilizan metadatos para definir permisos y roles, controlando quién tiene acceso a qué datos y qué operaciones pueden realizarse.

La combinación de datos y metadatos en las bases de datos relacionales proporciona un marco robusto para el almacenamiento, organización y gestión de la información, garantizando que los datos se mantengan consistentes, accesibles y utilizables a lo largo del tiempo.

1.3.2 Atributos y Entidades

En una base de datos relacional, los atributos son las características o propiedades que describen una entidad. Una entidad puede representar cualquier objeto del mundo real o conceptual sobre el cual se desea almacenar información. Por ejemplo, en una base de datos de un sistema de gestión de inventario, una entidad podría ser un producto, y los atributos podrían incluir el nombre del producto, su código, precio y cantidad en existencia.

Un atributo, tomado de manera aislada, no tiene un valor significativo, ya que no genera ni contiene información útil por sí mismo. Solo cuando los atributos se agrupan en una relación con una entidad, adquieren un significado relevante y contribuyen a la comprensión de un hecho o situación.

1.3.3 Normalización y Dependencias Funcionales

La normalización es un proceso fundamental en el diseño de bases de datos relacionales que busca eliminar redundancias y dependencias anómalas entre los datos. Este proceso consiste

en descomponer las tablas en estructuras más pequeñas y menos redundantes, siguiendo una serie de formas normales que garantizan la integridad de los datos.

Las dependencias funcionales son un concepto clave en la normalización. Una dependencia funcional se refiere a la relación entre dos conjuntos de atributos, en la que el valor de un atributo determina el valor de otro. La clave primaria de una tabla, por ejemplo, define una dependencia funcional completa, ya que identifica de manera única cada registro en la tabla.

El proceso de normalización incluye varias etapas:

- Primera forma normal (1NF): Elimina los grupos de datos repetidos y asegura que cada campo contenga solo un valor atómico.
- Segunda forma normal (2NF): Elimina las dependencias parciales de atributos no clave respecto a la clave primaria.
- Tercera forma normal (3NF): Elimina las dependencias transitivas entre atributos no clave.

1.3.4 Gestores de Bases de Datos Relacionales (SGBDR)

Un Sistema de Gestión de Bases de Datos Relacionales (SGBDR) es un software que permite la creación, administración y manipulación de bases de datos relacionales. Los SGBDR se encargan de garantizar que los datos sean almacenados de manera eficiente, accesible y segura, cumpliendo con las reglas establecidas por el modelo relacional.

El SGBDR más conocido es SQL, que se utiliza para realizar operaciones sobre los datos, como consultas, inserciones, actualizaciones y eliminaciones. Los SGBDR modernos también implementan las reglas de transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), que garantizan que las operaciones en la base de datos se realicen de manera fiable.

1.3.5 Las 12 Reglas de Codd

Edgar Codd propuso una serie de doce reglas para definir qué características debe cumplir un sistema para ser considerado verdaderamente relacional. Algunas de las reglas más importantes incluyen:

- Regla de la información: Toda la información debe representarse exclusivamente como valores en las tablas.
- Regla de acceso garantizado: Cada dato debe poder ser accesible de manera lógica mediante una combinación de su nombre de tabla, valor de clave primaria y nombre de columna.
- Tratamiento sistemático de los valores nulos: Los valores nulos deben gestionarse de manera uniforme y no deben confundirse con datos reales.
- Independencia lógica de los datos: Las modificaciones en el esquema lógico no deben afectar las aplicaciones existentes.
- Independencia física de los datos: Los cambios en la estructura de almacenamiento físico no deben afectar el acceso a los datos a nivel lógico.

Estas reglas han sido una guía para el desarrollo de los SGBDR, asegurando que los sistemas gestionen las bases de datos de manera coherente y eficiente.

1.3.6 Transacciones y Concurrencia

Los SGBDR también gestionan las transacciones que permiten agrupar una o varias operaciones en una unidad de trabajo que debe ejecutarse completamente o no ejecutarse en absoluto. Las transacciones en bases de datos relacionales siguen el principio ACID, que garantiza:

- Atomicidad: Todas las operaciones dentro de una transacción deben completarse. Si alguna falla, la transacción se deshace en su totalidad.
- Consistencia: Las transacciones llevan al sistema de un estado válido a otro estado válido, preservando las reglas definidas en el esquema de la base de datos.
- Aislamiento: Las transacciones concurrentes deben ejecutarse de manera que no interfieran entre sí.
- Durabilidad: Una vez que una transacción se completa, sus cambios son permanentes, incluso en caso de fallos del sistema.

La gestión de la concurrencia es otro aspecto crucial de los SGBDR, ya que permite que múltiples usuarios accedan y modifiquen los datos simultáneamente sin generar inconsistencias. Para manejar la concurrencia, se utilizan mecanismos como el bloqueo de registros y el control de versiones.

1.3.7 Resumen

El desarrollo de las bases de datos relacionales y sus gestores ha sido un punto de inflexión en la evolución de los sistemas de información. Desde su conceptualización por Edgar F. Codd en 1969, las bases de datos relacionales han proporcionado una estructura lógica y matemática que permite almacenar y manipular grandes volúmenes de datos de manera eficiente y segura. El uso de lenguajes de consulta como SQL y la implementación de las reglas de Codd han permitido la creación de sistemas de información robustos y escalables.

La teoría de conjuntos, las dependencias funcionales, la normalización y la lógica de predicados son conceptos fundamentales que sostienen el modelo relacional, garantizando la integridad y consistencia de los datos. Los sistemas gestores de bases de datos relacionales (SGBDR) implementan estos principios, proporcionando las herramientas necesarias para la administración efectiva de los datos en entornos empresariales y tecnológicos.

1.4 COMPETENCIAS Y HABILIDADES UTILIZADAS EN EL TFM

Este tipo de trabajos técnicos exige a los estudiantes la capacidad para abordar fundamentos del mundo empresarial, los procesos de negocio, la cadena de valor, y los sistemas de información como base de toda la información corporativa, la estrategia de innovación empresarial, ecosistemas, metodologías y herramientas, y los campos de innovación a lo largo de la cadena de valor. Características y aplicaciones de los sistemas de información. Los proyectos de dirección, creación y gestión de procesos que ayuden a la transformación digital de las empresas, debiendo estos de ser capaces de afrontar con determinación todas las fases que los componen, así como solucionar de forma eficiente y efectiva cualquier tipo de inconveniente que surja en el proceso. Por ello, durante la realización de este proyecto, se han desarrollado una serie de competencias que se enumeran a continuación;

1.4.1 Competencias básicas y generales:

- CB10. Que los estudiantes posean las habilidades de aprendizaje que les permitan continuar estudiando de un modo que habrá de ser en gran medida autodirigido o autónomo.
- CG2. Demostrar dominio en la utilización de bibliografía científica y bases de datos, así como en el análisis de documentos científico-técnicos, en el ámbito de la Transformación Digital de Empresas.
- CG4. Saber interpretar el marco normativo básico regulador del ámbito de la Transformación Digital de Empresas.

1.4.2 Competencias transversales:

- CT1. Mostrar compromiso con el respeto y promoción de los Derechos Humanos, la cultura de la paz y la conciencia democrática, los mecanismos básicos para la participación ciudadana y una actitud proactiva para la sostenibilidad ambiental y el consumo responsable.
- CT2. Examinar los Objetivos de Desarrollo Sostenible, especialmente los relacionados con la promoción del Estado de Derecho en los planos nacional e internacional; la garantía de acceso público a la información y proteger las libertades fundamentales, de conformidad con las leyes nacionales y los acuerdos internacionales; el fortalecimiento de las instituciones nacionales pertinente mediante la cooperación internacional, y la promoción de leyes y políticas no discriminatorias en favor del desarrollo sostenible.
- CT3. Aplicar la igualdad de género y la reducción de desigualdades en la sociedad a través del conocimiento y la educación y desarrollar un compromiso ético como ciudadano y como profesional.
- CT5. Desarrollar las aptitudes para el trabajo, la comunicación efectiva, la planificación y gestión del tiempo, el esfuerzo, el aprendizaje permanente, la búsqueda de la calidad, así como el espíritu creativo y emprendedor, además del liderazgo, para el adecuado desarrollo de proyectos académicos y profesionales.

1.4.3 Competencias especificas:

- CE15. Examinar las diferentes etapas que forman la cadena de valor del sector y sus mecanismos de control de calidad y evaluar las posibilidades de mejora de la eficiencia de sus procesos mediante la aplicación de metodologías habilitadoras de la transformación digital.
- CE16. Identificar, analizar e integrar las diferentes fuentes de información de datos generados en la empresa y aplicarlas al proceso de toma de decisiones.

1.4.4 Competencia específica del itinerario:

- CEI1. Saber analizar y proponer nuevas soluciones tecnológicas y de mejora en el ámbito del Sector energético.

1.4.5 Competencias para realizar el TFM:

- C01. Compara los servicios, los mecanismos y las herramientas de seguridad y de datos.
- COM1. Mostrar compromiso con el respeto y promoción de los Derechos Humanos, la cultura de la paz y la conciencia democrática, los mecanismos básicos para la participación ciudadana y una actitud proactiva para la sostenibilidad ambiental y el consumo responsable.

- COM2. Examinar los Objetivos de Desarrollo Sostenible, especialmente los relacionados con la promoción del Estado de Derecho en los planos nacional e internacional; la garantía de acceso público a la información y proteger las libertades fundamentales, de conformidad con las leyes nacionales y los acuerdos internacionales; el fortalecimiento de las instituciones nacionales pertinente mediante la cooperación internacional, y la promoción de leyes y políticas no discriminatorias en favor del desarrollo sostenible.
- COM3. Aplicar la igualdad de género y la reducción de desigualdades en la sociedad a través del conocimiento y la educación y desarrollar un compromiso ético como ciudadano y como profesional.
- COM4. Interpretar la información y aplicar el conocimiento de forma crítica.
- COM7. Identificar las principales amenazas en los diferentes campos de aplicación y evaluar y gestionar los riesgos asociados.
- COM8. Diferenciar y adaptar las herramientas, protocolos y plataformas de desarrollo de IoT.
- COM9. Examinar las diferentes etapas que forman la cadena de valor del sector y sus mecanismos de control de calidad y evaluar las posibilidades de mejora de la eficiencia de sus procesos mediante la aplicación de metodologías habilitadoras de la transformación digital.
- COM10. Identificar, analizar e integrar las diferentes fuentes de información de datos generados en la empresa y aplicarlas al proceso de toma de decisiones.
- COM11. Identificar y analizar los procedimientos técnicos y administrativos necesarios para la elaboración y puesta en marcha de proyectos de transformación digital de empresas del sector.

1.5 ESTRUCTURA DE LA MEMORIA

La presente memoria del TFM se organiza en siete capítulos principales, estructurados de manera lógica para guiar al lector a través de las diferentes etapas del proyecto, desde su conceptualización hasta su implementación y validación. A continuación, se describe brevemente el contenido de cada capítulo:

- Memoria Descriptiva del Proyecto: Este capítulo introduce el proyecto, detallando la motivación, los objetivos y el contexto en el que se desarrolla. Se explican las competencias y habilidades utilizadas durante el desarrollo del TFM, junto con un resumen de los resultados obtenidos.
- Fases de la Realización del TFM y su Cronograma Asociado: En este apartado se detallan las diferentes fases del proyecto, desde la definición inicial hasta la documentación final. Se presenta un cronograma que describe la duración y el orden de cada una de estas fases, estableciendo los hitos clave y el tiempo asignado a cada actividad.
- Implementación de la aplicación: Este capítulo profundiza en las especificaciones técnicas del sistema desarrollado. Se describen los detalles del diseño de la base de datos en MariaDB, el desarrollo de la aplicación en C#, y las herramientas utilizadas para su implementación. También se incluye el diseño de la arquitectura del sistema y la definición de los requisitos técnicos y funcionales.
- Formularios: Aquí se describen los diferentes formularios para el proceso CRUD de la base de datos aspectos generales del proyecto.,
- Documentación Gráfica: Este capítulo contiene la documentación gráfica que respalda el desarrollo del proyecto, como diagramas de entidad-relación, esquemas de la base de datos,

y otros elementos visuales que ayudan a comprender el diseño y la estructura del sistema. También se incluyen capturas de pantalla de la interfaz desarrollada y cualquier otro material gráfico relevante para el proyecto.
- Resultados y discusión: análisis económico, las normas y estándares de calidad aplicados, así como cualquier otra consideración relevante desde un punto de vista administrativo.

1.6 RESUMEN DE RESULTADOS

El TFM ha logrado cumplir satisfactoriamente con los objetivos planteados, diseñando e implementando una base de datos relacional que centraliza y gestiona de manera eficiente la información de ofertas y proyectos en la empresa AERTEC. El proyecto ha permitido automatizar procesos críticos, mejorando la trazabilidad y el control de los productos fabricados, desde su creación hasta su entrega al cliente final.

A continuación, se resumen los principales resultados obtenidos:

- Diseño e Implementación de la Base de Datos en MariaDB: Se ha desarrollado una base de datos relacional en MariaDB, siguiendo las mejores prácticas de diseño y normalización. El modelo de datos ha sido optimizado para minimizar la redundancia y asegurar la consistencia de la información. Las tablas y relaciones definidas permiten gestionar con precisión las piezas y componentes, como sus especificaciones y la documentación asociada, asegurando una trazabilidad completa de los productos.
- Automatización de Procesos: La implementación de esta base de datos ha permitido la automatización de varios procesos que anteriormente se gestionaban de manera manual mediante hojas de cálculo. Esta automatización ha reducido significativamente el riesgo de errores humanos, mejorado la eficiencia en la gestión de proyectos y facilitado la toma de decisiones informadas.
- Integración con la Infraestructura de AERTEC: El sistema diseñado ha sido configurado para operar inicialmente en entornos locales, permitiendo su validación. Aunque queda pendiente la aprobación de integración en los sistemas de la empresa.
- Desarrollo de Formularios en Visual Studio y C#: Se ha desarrollado un conjunto de formularios en Visual Studio utilizando C# para facilitar la interacción de los usuarios con la base de datos. Estos formularios ofrecen una interfaz intuitiva y amigable para la gestión de ofertas y proyectos, permitiendo a los usuarios realizar consultas, insertar nuevos registros y generar informes de manera eficiente.
- Validación y Pruebas: Se ha llevado a cabo un proceso exhaustivo de validación del sistema. Las pruebas realizadas han verificado la integridad y el rendimiento de la base de datos, así como la correcta implementación de las funcionalidades requeridas. Los usuarios finales han revisado la aplicación y confirmado su usabilidad y efectividad en la gestión diaria de proyectos.
- Trazabilidad y Gestión de Números de Serie: Uno de los resultados más importantes ha sido la implementación de la funcionalidad de inventariado de números de serie de los productos fabricados. Esta característica permite a la empresa llevar un control preciso de cada producto, mejorando la trazabilidad y facilitando la respuesta ante consultas de clientes o necesidades de mantenimiento.
- En este apartado se debe decir que en muchos casos la falta y concreción de los números de serie proviene del no cumplimiento de sus propios procedimientos del cliente, tema que se intenta cubrir desde AERTEC.

En conclusión, el proyecto ha logrado digitalizar y modernizar la gestión de información en AERTEC, mejorando la eficiencia operativa y la calidad del servicio. Los resultados obtenidos confirman que la solución desarrollada no solo es técnicamente viable, sino que también representa un paso significativo hacia la transformación digital de la empresa. La implementación de esta base de datos asegura una gestión más estructurada y eficaz, posicionando a AERTEC para enfrentar los desafíos futuros en la industria aeroespacial. Este proyecto solo es para un departamento que es sistemas no embarcados, con lo que es posible que si se afianza y demuestra su eficacia se pueda pasar a otros sistemas de producción.

2.FASES Y CRONOGRAMA.

El presente capítulo detalla las diferentes fases del proyecto de diseño e implementación de una base de datos relacional para la gestión de piezas y componentes de prueba para aeronaves en AERTEC. Cada fase representa un conjunto estructurado de actividades diseñadas para cumplir con los objetivos del proyecto de manera eficiente y organizada. El proceso de desarrollo se ha dividido en varias etapas, cada una con sus propios objetivos, actividades y resultados esperados. Estas etapas incluyen la definición del proyecto, el análisis de requisitos, el diseño conceptual y lógico, la implementación, las pruebas y validación, y finalmente la documentación del proceso. El cronograma asociado proporciona una visión clara del tiempo asignado a cada fase, asegurando una gestión adecuada de los recursos y el cumplimiento de los plazos establecidos. A continuación, se presenta una descripción detallada de cada una de estas fases y su cronograma, destacando las actividades clave y los resultados esperados en cada etapa del desarrollo del proyecto.

- Definición del Proyecto (1 semana): Estableciendo los objetivos del proyecto, definir el alcance y los requisitos de la base de datos, identificar las fuentes de datos y se planificar la metodología de trabajo.
- Análisis de Requisitos (2 semanas): Análisis detallado de los requisitos de la base de datos, incluyendo la identificación de entidades, atributos, relaciones y restricciones. Documentar los requisitos en un documento formal.
- Diseño Conceptual (3 semanas): Crear un modelo conceptual de la base de datos utilizando una herramienta de modelado, como Entidad-Relación.
- Diseño Lógico (2 semanas): Especificar las tablas, índices, claves primarias y foráneas, y se normaliza el diseño.
- Implementación (2 semanas): Implementar la base de datos según el diseño lógico. Crear las tablas y definir las relaciones, elaborar los scripts de creación de la base de datos y realizar pruebas preliminares para verificar la funcionalidad.
- Pruebas y Validación (1 semana): Realizar pruebas exhaustivas para validar la base de datos, incluyendo pruebas de integridad, rendimiento y seguridad. Corregir los errores y se ajusta el diseño según las necesidades.
- Documentación (1 semana): Documentar el proceso de desarrollo, elaborar un informe técnico que describa la estructura y funcionamiento de la base de datos, y preparar una presentación para exponer los resultados a los usuarios.

Figura 2.1 *Cronograma elaboración propia*

2.1. DEFINICIÓN DEL PROYECTO

La fase de definición del proyecto constituye el punto de partida para la planificación y organización de todo el proceso de desarrollo. En esta etapa inicial, se establecen los objetivos principales del proyecto, que son diseñar e implementar una base de datos que permita optimizar la gestión de ofertas y proyectos en AERTEC, garantizando una trazabilidad completa de los productos fabricados.

Durante esta fase, también se define el alcance del proyecto. Se delimita qué aspectos del proceso de gestión serán cubiertos por el sistema y cuáles quedarán fuera del alcance del desarrollo. En este contexto, el alcance incluye la gestión de piezas y componentes, especificaciones de fabricación y certificación, y la documentación asociada a cada producto.

Paralelamente, se identifican las fuentes de datos existentes en AERTEC y que podrían en un futuro integrar, por lo que convienen que la aplicación sea lo más abierta posible. Estos datos incluyen, entre otros, los registros de ofertas y proyectos en Excel, las especificaciones técnicas de los productos y la documentación asociada.

Finalmente, se planifica la metodología de trabajo a seguir. Se decide emplear una metodología en cascada (Figura 2.2), que permite ir avanzando, aunque en algún caso se deba de volver a redefinir parámetros.

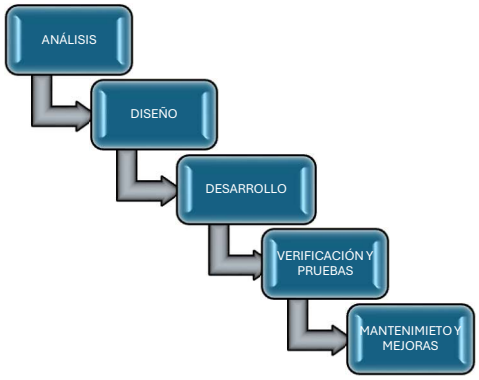


Figura 2.2 *Etapas de la metodología en cascada. Elaboración propia*

Aunque se determinen esas fases, algunas de estas se descomponen en varias tareas y se debe coordinar el avance síncrono de las mismas.

2.1.1. Establecimiento de objetivos

En esta fase se analizan con el cliente las necesidades de los usuarios finales de la aplicación y se acuerda los objetivos que se deben cubrir. Además, se especifican requisitos, incluyendo el control de lo solicitado por el cliente, fecha solicitada, fechas previstas de los materiales necesarios, y muchos campos que se deberán introducir en futuras ampliaciones de la aplicación.

El alcance del proyecto incluirá los campos de las dos tablas que existen actualmente, incluyéndose además pedidos, fechas de los pedidos y fecha de las ofertas.

2.1.2. Alcance y delimitación

El presente apartado define el alcance del proyecto, centrando los esfuerzos en áreas críticas que aseguren el cumplimiento de los objetivos principales establecidos por AERTEC. Se detallarán las funcionalidades a implementar en la base de datos para la gestión de ofertas y proyectos, trazabilidad de piezas y componentes, especificaciones de fabricación y certificación, mientras se excluyen integraciones complejas y se limitan las funcionalidades a lo esencial para una implementación exitosa dentro del plazo establecido. Esta delimitación precisa del alcance permitirá maximizar el impacto del proyecto, garantizando eficiencia y cumplimiento normativo en el manejo de datos y procesos internos.

- Gestión de Ofertas y Proyectos: Se diseñará una base de datos que permitirá gestionar de manera centralizada todas las ofertas y proyectos. Esto incluye la creación de tablas específicas que registrarán información clave, fechas de solicitud, fechas previstas de entrega de materiales, y estados de los proyectos. Además, se integrarán campos adicionales como fechas de pedidos y ofertas, que actualmente no están incluidos en el sistema Excel utilizado por AERTEC.
- Trazabilidad de Piezas y Componentes: Se incluirán funcionalidades que aseguren la trazabilidad completa de los productos fabricados desde su diseño hasta su entrega al cliente, con el número de serie (actualmente en muchos casos no aparecen). Esto permitirá un seguimiento detallado de cada pieza o componente, garantizando que se cumplan los requisitos técnicos y normativos.
- Especificaciones de Fabricación y Certificación: El sistema almacenará las especificaciones técnicas y la documentación asociada a cada producto, lo que permitirá una gestión más eficiente y segura de la información crítica para la fabricación y certificación de los componentes aeronáuticos.
- Exclusión de Integraciones Complejas: El proyecto se centrará en la implementación de una base de datos que funcione inicialmente en un entorno local. No se abordarán en esta fase integraciones complejas con otros sistemas de la empresa, como ERP (Planificación de Recursos Empresariales) o sistemas de gestión documental, aunque se considerará la posibilidad de futuras ampliaciones.
- Limitación a Funcionalidades Básicas: Aunque se ha identificado la necesidad de incluir más campos y funcionalidades en el futuro, el desarrollo actual se centrará en las tablas y datos existentes, y algunos campos adicionales básicos. Esto garantiza que el proyecto pueda ser realizado y completado dentro del plazo acordado.
- Reducción del Alcance a Productos Actuales: La implementación inicial se limitará a los productos y procesos actualmente gestionados por el departamento de piezas y componentes de pruebas para equipos no embarcables (son aquellos que no están diseñados para ser instalados ni utilizados a bordo de una aeronave durante el vuelo. Estos equipos se utilizan principalmente en operaciones de soporte en tierra o en procesos relacionados con el mantenimiento, reparación, pruebas o configuración de la aeronave, pero no son adecuados o necesarios para su operación en vuelo), sin abordar expansiones hacia nuevas líneas de productos o mercados, que podrán considerarse en futuras fases de desarrollo.

Diferencias clave entre los equipos embarcables y otros es que los primeros deben cumplir con estrictas normativas y certificaciones de seguridad aeronáutica, mientras que los demás no requieren estas certificaciones, ya que no están expuestos a las condiciones de vuelo.

En cuanto a la funcionalidad en vuelo los equipos embarcables tienen funciones que son críticas para la operación segura y eficiente de la aeronave en vuelo, mientras que los otros se utilizan para soporte en tierra o para preparar la aeronave antes del vuelo.

Esta delimitación clara y precisa del alcance del proyecto permitirá que se concentren los esfuerzos en las áreas de mayor impacto, garantizando que los objetivos establecidos se cumplan de manera eficiente y dentro de los plazos estipulados. Para realizar pruebas se cogerán un número menor a lo inventariado actualmente para que sea más manejable,

2.1.3. Identificación de fuentes de datos

Las fuentes de datos suministrada por el cliente han sido dos ficheros de Excel, uno denominado como Project list (listado de proyectos) (Figura 2.3) en el cual se almacenan los diferentes proyectos y los responsables de estos.

Codigo_Proyecto	Estado	Oferta	Cliente	Titulo	Specific_QAP	Jefe_Proyecto	WP_Hardwa	WP_Manufa	WP_Softwar	SW repository	Repositorio	KPIs	Motivo
18-P006	Activo	(Manufacturing)	Airbus Transversal	CATS	NO	Antonio Fajardo	NO	NO	YES	Airbus CATS	Airbus CATS	NO	
18-P006ECD	No Iniciado	(Manufacturing)	Airbus Transversal	CATS - Export Control	NO	Antonio Fajardo	NO	NO	YES	Airbus CATS	Airbus CATS	NO	KPIs especificos de Airbus Manufacturing
19-P002	Activo	varios	Sopra	DMAT	NO	Francisco Carmona	NO	NO	NO	Airbus DMAT	Airbus DMAT	NO	No requerido
20-P033	Cerrado	19-Of-185	Pegasus	MONIF	NO	Francisco Carmona	NO	NO	YES	TFS	N/A	NO	No requerido
21-P062	Cerrado	20-Of-601	Indra	Banco Cableado	NO	Francisco Carmona	YES	YES	YES	TFS	DMS	NO	
21-P073	Cerrado	N/A	CDTI	ONeIRE	NO	Francisco Carmona	NO	NO	YES	TFS	DMS	NO	
22-P033	Cerrado	22-Of-504	Airbus LTA	MLG-AIM	YES	Sergio Zambrano	YES	YES	YES	TFS	DMS	NO	
22-P051	Cerrado	22-Of-453	Airbus A330	Hydraulic Test bench	YES	Francisco Carmona	YES	YES	NO	N/A	DMS	NO	
22-P65	Cerrado	22-Of-729	Airbus A400M	A400M grupo electrógeno	NO	Francisco Carmona	YES	YES	NO	N/A	DMS	NO	

Figura 2.3 *Listado de Proyectos. Elaboración propia*

El otro archivo denominado Ítems fabricados, (Figura 2.4), los números de serie de las piezas y su denominación, en él también se incluyen algunos campos nuevos y también un número de serie dado por AERTEC en algunos casos.

AERTEC — ITEMS FABRICADOS

OFERTA AERTEC	CLIENTE	CÓDIGO PROTEC	DENOMINACIÓN PROTEC	UBICACIÓN PROTECTO	ESPECIFICACIÓN	PN AERTEC / FABRICANTE	PN CLIENTE	TIPO	PROGRAMA	UNIDADES S/N	Certificados CoC	MES / AÑO (ENTREGA)	CHECKLIST AERTEC (ENTREGA)
23-Of-092_090	ADS	23-P116	CHECKOUTERS KIT MODIFICATIONS	23-P116 - CHECKOUTERS KIT MODIFICATIONS	TAO-FII-SP-23-0022	KIT 02: K02-MH1-03-03-YUTH-02-THALES RADIOS HARNESS BoB; KIT 03: K03-MH1-13-ESIU-ESIU HARNESS BoB; KIT 10: K10-MH1-03-03-YUTH-02-THALES RADIOS HARNESS BoB; KIT 12: K12-MH1-02-SDR-SDR HARNESS BoB; KIT 22: K22-MH1-04-ICU_ICUS HARNESS BoB	KIT 02: K02-MH1-03-03-YUTH-02-THALES RADIOS HARNESS BoB; K02-MH1-13-SAM-SAM HARNESS BoB; K02-MH1-14-AUDIOISE-AUDIO MIXER HARNESS BoB; K02-MH1-15-TGU-TGU HARNESS BoB; KIT 03: K03-MH1-05-02-ETH-ETHERNET SWITCH HARNESS BoB; K03-MH1-13-ESIU-ESIU HARNESS BoB; K03-MH1-14-DLIP-DLIP HARNESS BoB; KIT 10: K10-MH1-03-03-YUTH-02-THALES RADIOS HARNESS BoB; KIT 12: K12-MH1-02-SDR-SDR HARNESS BoB; KIT 22: K22-MH1-04-ICU_UAE2-ICUS HARNESS BoB	Mixed Consols	HRTT	10			

Ref. documento: Kit-09-S01-
Fecha actuali.: 20/08/2024
Fecha Or.: 16/11/2017

Figura 2.4 *Listado de Items Fabricados. Elaboración propia*

Este fichero se encuentra en red y es completado habitualmente por los responsables de los proyectos por lo que en algunos momentos se crea conflictos de incompatibilidades, y es uno de los objetivos principales del trabajo.

El otro también muy importante es que en la misma celda del número de serie se suele introducir la descripción de estos y algunas aclaraciones, (Figura 2.5), para comprender unas de las cuestiones a tratar en la normalización de datos.

PN AERTEC / FABRICANTE ver part list	PN CLIENTE
KIT 02:	**KIT 02:**
K02-MH1-03-03-VUTH-02-THALES RADIOS HARNESS BoB	**K02-MH1-03-03-VUTH-02-THALES RADIOS HARNESS BoB**
23P110-MAZ CHECK KIT2-M01-A 001	PRAV-01-4300-11014-001-A
K02-MH1-13-SAM-SAM HARNESS BoB	**K02-MH1-13-SAM-SAM HARNESS BoB**
23P110-MAZ CHECK KIT2-M02-A 001	PRAV-01-4300-11015-001-A
K02-MH1-14-AUDMIX-AUDIO MIXER HARNESS BoB	**K02-MH1-14-AUDMIX-AUDIO MIXER HARNESS BoB**
23P110-MAZ CHECK KIT2-M03-A 001	PRAV-01-4300-11016-001-A
K02-MH1-15-TGU-TGU HARNESS BoB	**K02-MH1-15-TGU-TGU HARNESS BoB**
23P110-MAZ CHECK KIT2-M04-A 001	PRAV-01-4300-11017-001-A
KIT 03:	**KIT 03:**
K03-MH1-05-02-ETH-ETHERNET SWITCH HARNESS BoB	**K03-MH1-05-02-ETH-ETHERNET SWITCH HARNESS BoB**
23P110-MAZ CHECK KIT3-M01-A 001	PRAV-01-4600-10613-001-A
K03-MH1-13-ESIU-ESIU HARNESS BoB	**K03-MH1-13-ESIU-ESIU HARNESS BoB**
23P110-MAZ CHECK KIT3-M02-A 001	PRAV-01-4600-10614-001-A
K03-MH1-14-DLIP-DLIP HARNESS BoB	**K03-MH1-14-DLIP-DLIP HARNESS BoB**
23P110-MAZ CHECK KIT3-M03-A 001	PRAV-01-4600-10615-001-A
KIT 10:	**KIT 10:**
K10-MH1-03-03-VUTH-02-THALES RADIOS HARNESS BoB	**K10-MH1-03-03-VUTH-02-THALES RADIOS HARNESS BoB**
23P110-MAZ CHECK KIT10-M01-A 001	PRAV-01-4300-11106-001-A
KIT 13:	**KIT 13:**
K13-MH1-02-SDR-SDR HARNESS BoB	**K13-MH1-02-SDR-SDR HARNESS BoB**
23P110-MAZ CHECK KIT13-M01-A 001	PRAV-01-3137-10002-001-A
KIT 22:	**KIT 22:**
IK22-MH1-04-ICU_ICUS HARNESS BoB	**K22-MH1-04-ICU_UAE2-ICUS HARNESS BoB**
23P110-MAZ CHECK KIT22-M01-A 001	PREL-01-2450-10103-001-A

Figura 2.5 *Listado de Items Fabricados detalle. Elaboración propia*

A estas columnas de datos se deberán anexar dos más con la descripción y un aclaratorio, Se puede ver detalle en la (Figura 2.6). Al ser rellenado por cada responsable no hay un criterio común para su implementación. Coincidiendo con este proyecto se ha elaborado una instrucción técnica para normalizar este tema IT09.S01 aprobada el 4/4/2024, pero pendiente de su implementación en estos momentos.

CÓDIGO PROYECT	PN AERTEC / FABRICANTE ver part list	PN CLIENTE	DESCRIPCION	ACLARATORIO
23-P110	23P110-MAZ CHECK KIT2-M01-A 001	PRAV-01-4300-11014-001-A	K02-MH1-03-03-VUTH-02-THALES RADIOS HARNESS BoB	KIT 02:
23-P110	23P110-MAZ CHECK KIT2-M02-A 001	PRAV-01-4300-11015-001-A	K02-MH1-13-SAM-SAM HARNESS BoB	KIT 02:
23-P110	23P110-MAZ CHECK KIT2-M03-A 001	PRAV-01-4300-11016-001-A	K02-MH1-14-AUDMIX-AUDIO MIXER HARNESS BoB	KIT 02:
23-P110	23P110-MAZ CHECK KIT2-M04-A 001	PRAV-01-4300-11017-001-A	K02-MH1-15-TGU-TGU HARNESS BoB	KIT 02:
23-P110	23P110-MAZ CHECK KIT3-M01-A 001	PRAV-01-4600-10613-001-A	K03-MH1-05-02-ETH-ETHERNET SWITCH HARNESS BoB	KIT 03:
23-P110	23P110-MAZ CHECK KIT3-M02-A 001	PRAV-01-4600-10614-001-A	K03-MH1-13-ESIU-ESIU HARNESS BoB	KIT 03:
23-P110	23P110-MAZ CHECK KIT3-M03-A 001	PRAV-01-4600-10615-001-A	K03-MH1-14-DLIP-DLIP HARNESS BoB	KIT 03:
23-P110	23P110-MAZ CHECK KIT10-M01-A 001	PRAV-01-4300-11106-001-A	K10-MH1-03-03-VUTH-02-THALES RADIOS HARNESS BoB	KIT 10:
23-P110	23P110-MAZ CHECK KIT13-M01-A 001	PRAV-01-3137-10002-001-A	K13-MH1-02-SDR-SDR HARNESS BoB	KIT 13:
23-P110	23P110-MAZ CHECK KIT22-M01-A 001	PREL-01-2450-10103-001-A	K22-MH1-04-ICU_UAE2-ICUS HARNESS BoB	KIT 22:

Figura 2.6 *Desglose de Items, Descripción y Aclaratorio. Elaboración propia*

2.2. ANÁLISIS DE REQUISITOS

Para la apertura de un proyecto es imprescindible que tenga asociado pedido y oferta.

Los números de serie de las piezas y componentes son unívocos.

Procedimientos e instrucciones que se deben de considerar:

- Procedimientos:
  - PG09 Diseño y desarrollo (última versión).
  - PG13_Control de documentos (última versión).
- Instrucciones Técnicas:
  - IT.09.S03 Gestión de Proyectos DOA (última versión).
  - IT.10.01 Proyectos de Fabricación (última versión).

  - IT.10.S01 Proyectos de Fabricación
- Documentación:
  - ISO 9001 (última versión)
  - ISO 9100 (última versión)
  - DP-000-097 A Concesiones de Proveedores (última versión).

Resumen del la IT.10 etiquetado del producto, número de pieza. A todos los equipos en fase de diseño se les debe asignar un código interno conocido como PN. Una vez finalizada tanto la fase de diseño como la de fabricación, se asignará el S/N del equipo, según las siguientes indicaciones.

El código interno de los productos de AERTEC será el siguiente:

- PN: Nombre- subnombre -xx- yy -zzz
- Nombre: designado por el líder del proyecto.
- Subnombre: Nombre de los subconjuntos designados por el líder del proyecto.
- xx: el número secuencial asignado a cada elemento diseñado del conjunto de artículos de una lista de materiales de un producto.
- yy: el número secuencial asignado a cada versión de hardware o firmware que no cambia la lista de materiales.
- zzzz: el número secuencial asignado a cada lanzamiento de lote de fabricación del producto.

En los casos en que el cliente requiera una PN propia, ésta deberá constar en el expediente general de proyectos de fabricación del área. En este fichero se registrará el PN o código de especificación del cliente y el PN interno de AERTEC para tener trazabilidad entre PN cliente - PN interno de AERTEC. El PN del cliente estará etiquetado en el equipo según se especifica junto con el PN interno de la organización. El PN de este registro será realizado por el Gerente de Proyecto o por la persona en quien él delegue.

Para el buen funcionamiento del sistema y del procedimiento es conveniente se realicen auditorias periódicas para verificar el cumplimiento de estos requerimientos.

También se deben tener en cuenta las normas más importantes en la industria aeroespacial relacionadas con el desarrollo de sistemas y la evaluación de seguridad. (Anexo1).

2.2.2. Documentación de requisitos

En esta etapa, se detallan de manera clara y precisa los requisitos funcionales y no funcionales que el sistema debe cumplir, incluyendo especificaciones técnicas, restricciones operativas y criterios de calidad. Asegurando que el desarrollo del sistema se alinee con las expectativas del cliente y minimizando el riesgo de malentendidos. En este caso se definen tres:

- Gestión de (PN): La base de datos debe permitir la creación, almacenamiento y consulta de (PN) de los productos. Cada PN debe estar vinculado a sus especificaciones técnicas, asegurando una trazabilidad completa desde la solicitud hasta la entrega.
- Seguridad de Acceso: El sistema debe implementar controles de acceso basados en roles para garantizar que solo el personal autorizado pueda modificar o eliminar registros críticos en la base de datos. Las operaciones de consulta deben estar disponibles para todos los usuarios autorizados, mientras que las operaciones de escritura deben estar restringidas a ciertos roles específicos.

- Compatibilidad con Excel: La base de datos debe ser capaz de exportar datos hacia archivos Excel, permitiendo que los usuarios continúen utilizando sus herramientas actuales mientras se integra gradualmente el nuevo sistema. Esta compatibilidad debe incluir la importación de registros de ofertas y proyectos existentes.

2.3. DISEÑO CONCEPTUAL

El diseño conceptual es una etapa fundamental en el desarrollo de una base de datos, ya que permite definir de manera abstracta y estructurada la información que será gestionada por el sistema. En esta fase, se identifican las entidades clave, los atributos asociados a cada entidad y las relaciones entre estas. Este modelo conceptual no está limitado por las restricciones técnicas del software, sino que se enfoca en una representación clara y comprensible de la estructura de los datos. Para esta parte se ha utilizado los apuntes recibidos en las clases de Manuel Enciso y usando las aplicaciones de ORACLE Data Modeler y Developer.

2.3.1. Identificación de Entidades.

En el contexto del proyecto, las entidades representan los elementos principales que serán gestionados por la base de datos. Para la gestión de piezas y componentes de prueba en aeronaves, las entidades clave identificadas son las siguientes:

- Proyectos: Representa los proyectos que AERTEC maneja, cada uno asociado a la fabricación o mantenimiento de aeronaves específicas. Cada proyecto puede estar relacionado con múltiples ofertas y productos.
- Ofertas: Contiene información sobre las ofertas que se realizan para proyectos específicos, incluyendo detalles como la fecha de la oferta, el costo estimado, y la descripción de los productos o servicios ofertados.
- Productos: Representa los diferentes productos o componentes fabricados o gestionados en cada proyecto. Incluye detalles técnicos como números de parte (PN), especificaciones de fabricación, y certificaciones.
- Pedidos: Almacena los pedidos realizados para productos específicos, incluyendo la fecha del pedido, cantidad solicitada, y el estado del pedido.

2.3.2. Atributos de las Entidades.

Cada entidad se compone de varios atributos que describen sus características. A continuación, se detallan algunos de los atributos más importantes:

- Numero de Oferta/Proyecto. Este atributo representa un identificador único para cada oferta o proyecto gestionado en la base de datos. Es fundamental para identificar de manera única cada oferta o proyecto, evitando duplicidades y permitiendo la asociación correcta de información relacionada, como las fechas de inicio y fin, los clientes asociados, y el estado del proyecto. Sin un identificador único, la trazabilidad y la gestión de datos serían confusas y propensas a errores.
- Fecha de Solicitud y Fecha de Entrega. Atributos que registran la fecha en la que se solicitó una oferta o proyecto y la fecha prevista de entrega de materiales o productos finales. Permiten el seguimiento del tiempo de respuesta y cumplimiento de plazos, lo cual es crucial para la planificación y control de los proyectos. Estas fechas también facilitan la gestión de

inventarios y la programación de recursos, ayudando a optimizar la cadena de suministro y las operaciones logísticas.

- Estado del Proyecto. Indica el estado actual de un proyecto u oferta, como "En proceso", "Completado", "Cancelado", etc. Proporciona una visión clara y actualizada del progreso de cada proyecto, lo cual es esencial para la toma de decisiones estratégicas. Además, facilita la comunicación interna y con los clientes, asegurando que todas las partes interesadas estén al tanto del estado actual de los proyectos.
- Numero de Pieza/Componente. Identificador único para cada pieza o componente registrado en la base de datos. Fundamental para la trazabilidad completa de cada pieza o componente desde su fabricación hasta su entrega. Este atributo permite el seguimiento detallado de cada elemento, garantizando que se cumplan los requisitos técnicos y normativos y que se pueda identificar rápidamente cualquier pieza en caso de requerirse mantenimiento o revisión.
- Repositorio Especificaciones Técnicas, Documentación Asociada. Atributó que almacena la información técnica específica de cada pieza o componente, como materiales, dimensiones, y requisitos de rendimiento. Crítico para garantizar que los componentes cumplan con los estándares de calidad y las especificaciones del cliente. Facilita la gestión de calidad y certificación de los productos, asegurando que todas las piezas cumplen con las normativas y requisitos técnicos necesarios.
- Fecha de Pedido y Fecha de Oferta. Atributos que indican cuándo se realizó un pedido y cuándo se hizo una oferta. Estos atributos son esenciales para el seguimiento de la gestión comercial, permitiendo analizar el ciclo de ventas y la efectividad de las ofertas realizadas. Además, facilitan la programación de recursos y la gestión de inventarios, asegurando que los materiales necesarios estén disponibles a tiempo.

2.3.3. Relaciones entre Entidades.

El diseño conceptual también incluye la definición de las relaciones entre las entidades, que muestran cómo interactúan entre sí dentro del sistema. Las principales relaciones identificadas son:

- Relación Proyecto-Oferta: Un proyecto puede estar asociado a múltiples ofertas, pero cada oferta pertenece a un solo proyecto (relación uno a muchos).
- elación Proyecto-Producto: Un proyecto puede incluir múltiples productos, pero cada producto pertenece a un solo proyecto (relación uno a muchos).
- Relación Producto-Pedido: Un producto puede ser solicitado en varios pedidos, y un pedido puede incluir varios productos (relación muchos a muchos).
- Relación Producto-Especificación Técnica: Cada producto puede tener múltiples especificaciones técnicas asociadas, y cada especificación se aplica a un solo producto (relación uno a muchos).
- Relación Producto-Documentación: Un producto puede tener múltiples documentos asociados, como certificados o manuales, y cada documento está relacionado con un solo producto (relación uno a muchos).

2.3.4. Diagrama Entidad-Relación (ER).

Para representar visualmente el diseño conceptual, se utiliza un Diagrama Entidad-Relación (ER). Este diagrama muestra las entidades, sus atributos principales, y las relaciones entre ellas.

El diseño conceptual establece una base sólida para la posterior fase de diseño lógico y físico. Este diseño conceptual es el primer paso hacia la implementación de una base de datos eficiente que garantizará la trazabilidad, integridad y disponibilidad de la información crítica para la gestión de ofertas y productos en AERTEC.

2.3.5. Herramientas de modelado utilizadas

Las aplicaciones del software cliente Oracle Data Modeller; y Oracle SQL Developer del Departamento de Lenguajes y Ciencias de la Computación de la UMA.

2.4. DISEÑO LÓGICO

El diseño lógico consiste en transformar el modelo conceptual de la base de datos en un conjunto de estructuras que puedan ser implementadas en un sistema de gestión de bases de datos relacional, como MariaDB, garantizando la eficiencia y consistencia de los datos.

2.4.1. Tablas y Atributos

Las entidades identificadas en el diseño conceptual se convierten en tablas en el diseño lógico. A continuación, se describen las tablas principales y sus respectivos atributos (Figura 2.7)

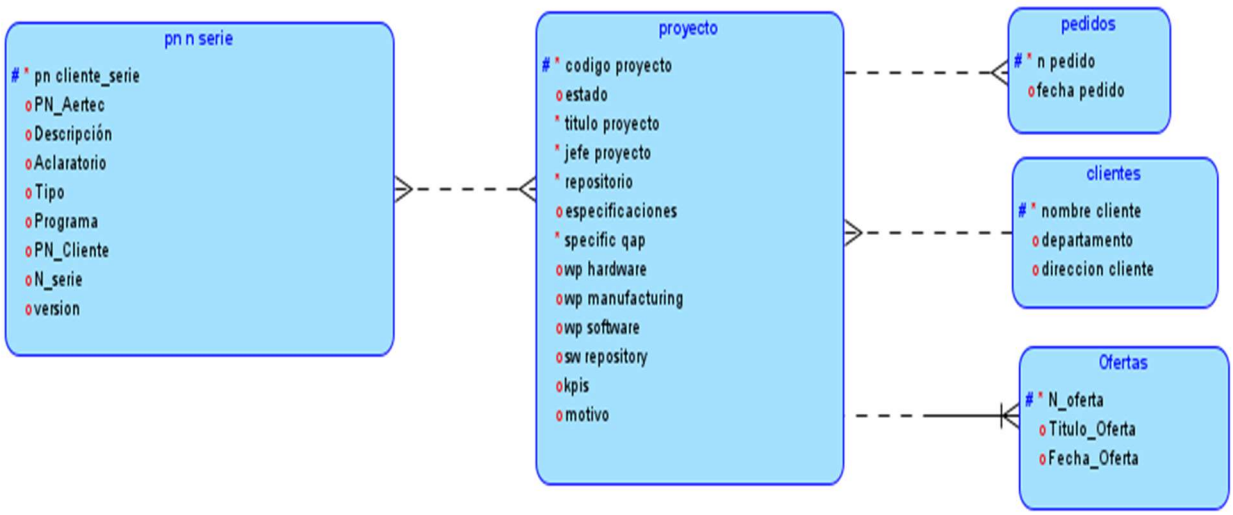


Figura 2.7 *Modelo Lógico. Elaboración propia*

En el proceso de diseño de bases de datos, el paso del modelo conceptual al modelo lógico implica la transformación de entidades en tablas, que representan las estructuras fundamentales para almacenar los datos. Cada tabla se compone de un conjunto de atributos, que son las columnas que definen las características de los datos que se registrarán. Estos atributos se derivan de las propiedades identificadas durante la fase de análisis y son esenciales para garantizar la integridad y consistencia de la información. A continuación, se presentan las tablas principales y sus respectivos atributos (Tablas 2.1 a 2.5)

Tabla 2.1 Entidad proyecto y sus atributos

Proyectos
Codigo_Proyecto
estado
título proyecto
jefe proyecto
repositorio
especificaciones
specific_qap
wp_hardware
wp_manufacturing
wp_software
sw_repository
KPIs
motivo

Tabla 2.1 *Entidad proyecto y sus atributos Elaboración Propia*

Tabla 2.2 Entidad Pedidos y sus atributos

Pedidos
Numero pedido
fecha pedido

Tabla 2.2 *Entidad Pedidos y sus atributos Elaboración Propia*

Tabla 2.3 Entidad PN Pieza- y sus atributos

P/N Pieza-componente
pn_cliente_serie
pn_aertec
descripción
aclaratorio
tipo
programa
PN cliente
Número de serie
versión

Tabla 2.3 *Entidad PN y sus atributos Elaboración Propia*

Tabla 2.4 Entidad Clientes y sus atributos

Clientes
nombre cliente
departamento
Dirección cliente

Tabla 2.4 *Entidad Clientes sus atributos Elaboración Propia*

Tabla 2.5 Entidad Ofertas y sus atributos

Ofertas
Numero oferta
Título oferta
Fecha oferta

Tabla 2.5 *Entidad Ofertas y sus atributos Elaboración Propia*

2.4.2. Relaciones y Claves Foráneas

Las relaciones entre las tablas se implementan utilizando claves foráneas (FK). Estas aseguran la integridad referencial y mantienen la consistencia de los datos.

- Relación; Proyecto-Oferta: La tabla Ofertas incluye una clave foránea ID_Proyecto que se relaciona con la clave primaria en la tabla Proyectos. Esto establece una relación uno a muchos entre Proyectos y Ofertas.
- Relación; Proyecto-Producto: La tabla Productos incluye una clave foránea ID_Proyecto que se relaciona con la clave primaria en la tabla Proyectos. Esto establece una relación uno a muchos entre Proyectos y Productos.
- Relación; Producto-Pedido: La tabla Pedidos incluye una clave foránea Numero_Parte que se relaciona con la clave primaria en la tabla Productos. Esto establece una relación uno a muchos entre Productos y Pedidos.

2.4.3. Normalización

Para garantizar que el diseño lógico es eficiente y libre de redundancias, se aplican técnicas de normalización. El diseño lógico debe cumplir con al menos la Tercera Forma Normal (3NF) para evitar problemas de redundancia y dependencia funcional.

Para mejorar el rendimiento de las consultas, se deben definir índices en los campos más frecuentemente utilizados en las búsquedas y en las claves foráneas. Tipos:

- Índices Primarios: Los índices se crean automáticamente en las claves primarias de cada tabla.
- Índices Secundarios: Se pueden crear índices adicionales en campos como Fecha_Pedido, Nombre_Producto, o Cliente para acelerar las consultas más comunes.

2.4.4. Consideraciones para la Integridad y Seguridad

Además de la estructura de las tablas y relaciones, es fundamental considerar la integridad de los datos y la seguridad.

Restricciones de Integridad: Se implementarán restricciones para asegurar que los datos sean válidos. Por ejemplo, las fechas no pueden ser nulas, y los campos numéricos como la cantidad de un pedido deben ser mayores que cero.

Permisos de Usuario: Se definirán roles y permisos en el DBMS para asegurar que solo los usuarios autorizados puedan realizar operaciones críticas, como la modificación o eliminación de registros.

2.4.5. Diseño Lógico

El diseño lógico, una vez completado y validado, servirá como base para la fase de implementación. Las estructuras lógicas definidas se traducirán directamente en scripts SQL para la creación de las tablas, claves, índices y relaciones en el DBMS seleccionado (MariaDB).

2.4.6. Modelo Relacional con Ingeniería de Data Modeler (Oracle)

Una vez revisado y realizado varias pruebas de la aplicación se ha llegado a la conclusión de que está acorde con las necesidades y que no da error de ejecución (véase Figura 2.8. donde está el modelo relacional para la base de datos).

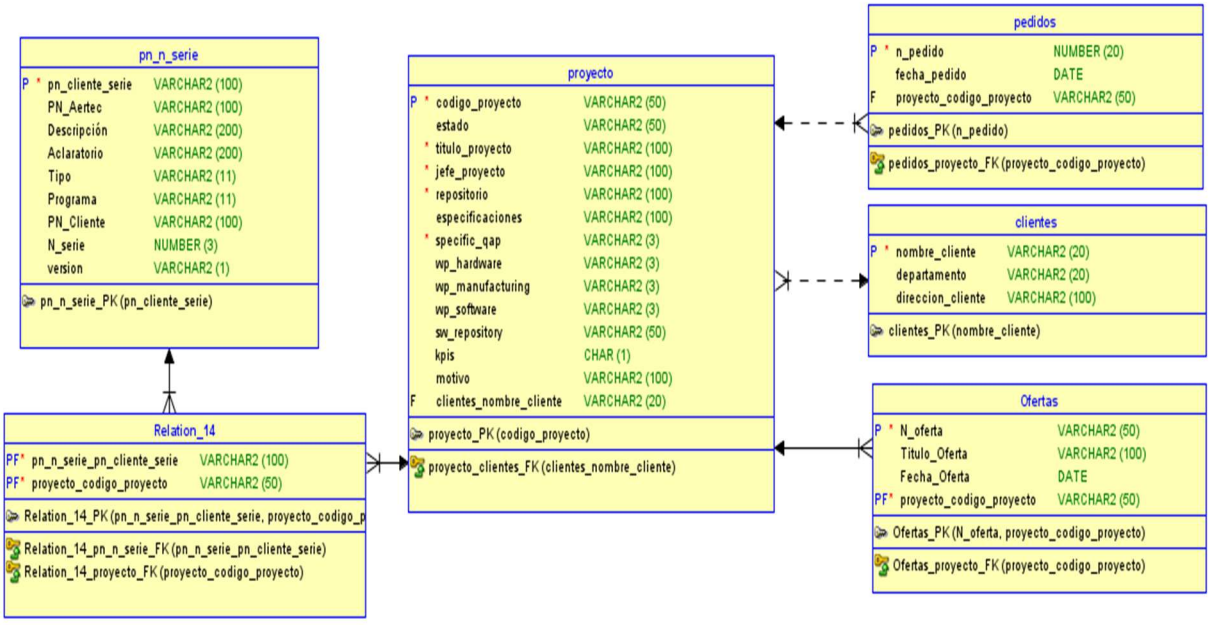


Figura 2.8 *Modelo Relacional. Elaboración propia*

En el Anexo 2.1 se puede ver el código generado automáticamente. Con él una vez importado al Developer se puede comprobar las tablas, relaciones creadas y restricciones (Figuras 2.9, a 2.11).

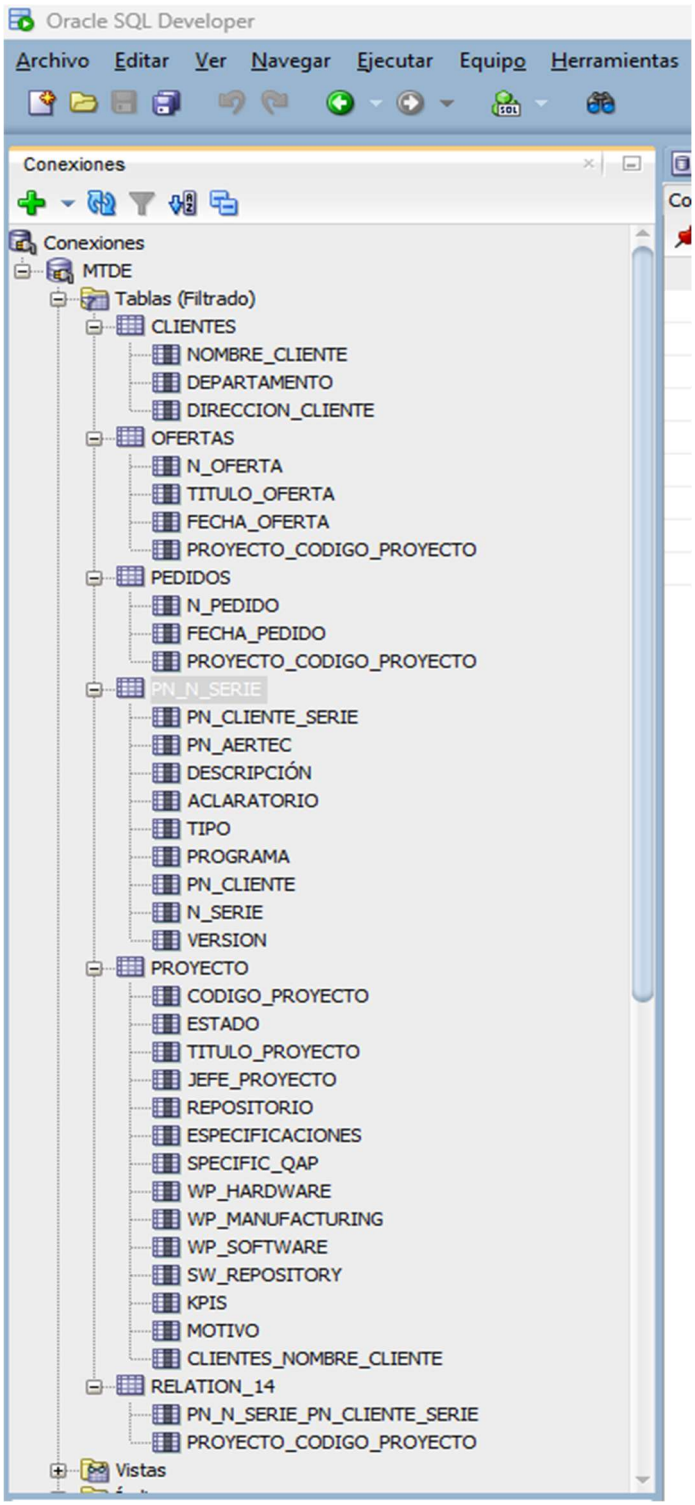


Figura 2.9 *Detalle de tablas generadas en Data Modeler. Elaboración propia*

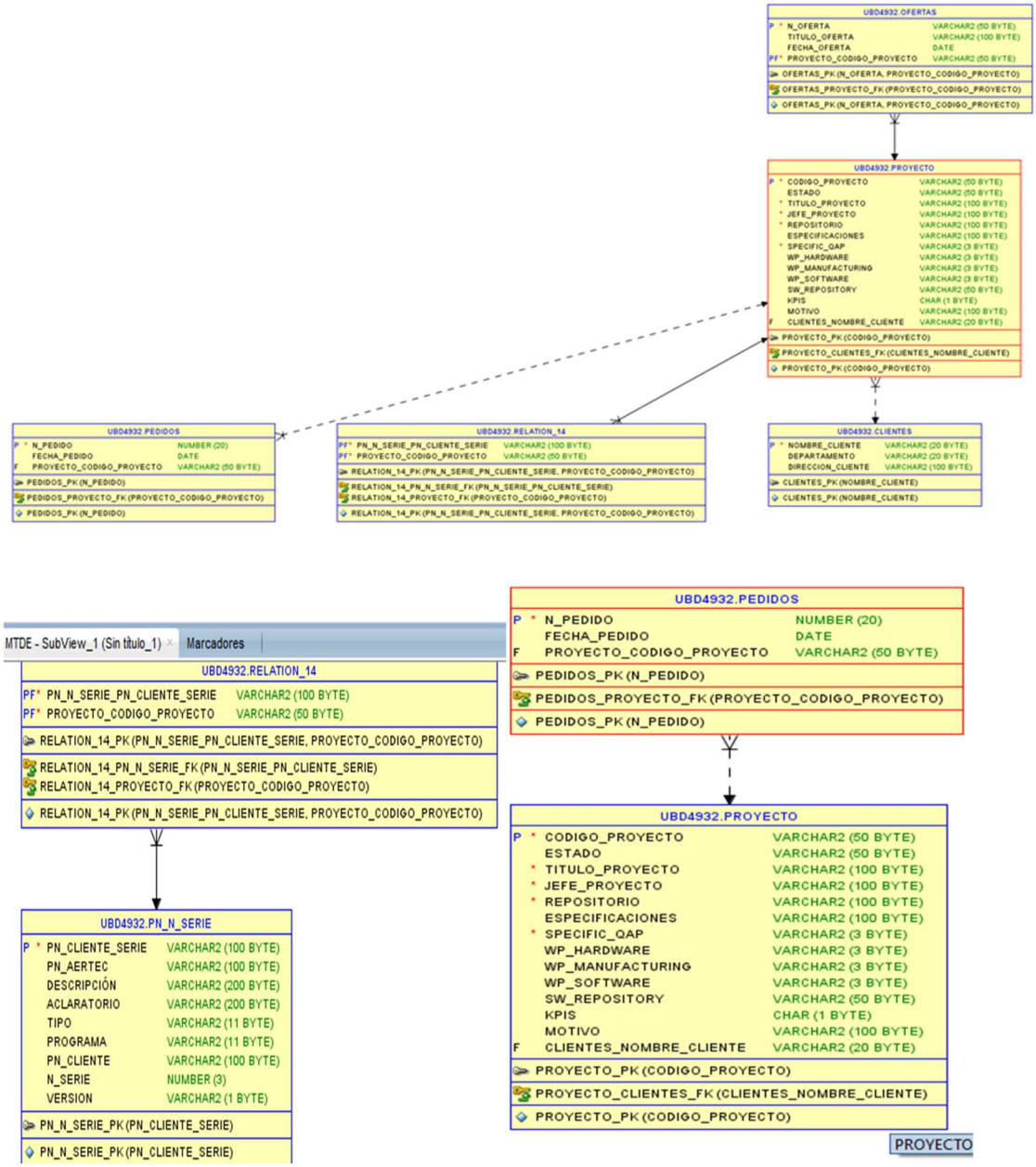


Figura 2.10 *Detalle de Relaciones generadas en Data Modeler. Elaboración propia*

Figura 2.11 *Detalle de Restricciones generadas en Data Modeler. Elaboración propia*

3. IMPLEMENTACIÓN DE LA APLICACIÓN CRUD CON C.# Y MARIADB

Una vez diseña para la base de datos se instala el software de MaríaDB [8] en local y se configura el servidor además de los usuarios iniciales, al igual que el puerto de comunicación. Para esta parte se ha utilizado HeidiSQL que es una herramienta gratuita y de código abierto utilizada para gestionar y administrar bases de datos. Proporciona una interfaz gráfica que facilita a los usuarios realizar diversas tareas de administración de bases de datos, como:

- Conexión y Gestión de Bases de Datos: Permite conectarse a servidores de bases de datos remotos o locales, gestionar varias conexiones simultáneamente, y navegar por las estructuras de las bases de datos (tablas, vistas, procedimientos almacenados, etc.).
- Consulta SQL: Ofrece un editor de SQL donde se puede escribir, ejecutar y depurar consultas SQL. También permite visualizar los resultados de las consultas y exportarlos a diferentes formatos (como CSV, XML, HTML).
- Diseño y Modificación de Tablas: Facilita la creación, edición y eliminación de tablas y sus columnas, índices, claves primarias y foráneas, todo a través de una interfaz gráfica intuitiva.
- Importación y Exportación de Datos: HeidiSQL permite importar y exportar datos entre diferentes formatos, incluyendo CSV, SQL y otros. Esto es útil para realizar respaldos de bases de datos o migrar datos entre servidores.
- Gestión de Usuarios y Permisos: Los usuarios pueden gestionar cuentas de usuarios y configurar los permisos de acceso a las bases de datos, asegurando que solo los usuarios autorizados puedan realizar ciertas acciones.
- Visualización de Esquemas y Relaciones: Permite visualizar y gestionar relaciones entre tablas. Aunque no es su mejor característica. Es una herramienta eficiente y fácil de usar para gestionar bases de datos. Su popularidad se debe a su simplicidad, su enfoque en la facilidad de uso, y la riqueza de funciones que ofrece de manera gratuita.

3.1. CREACIÓN DE USUARIO EN MARIADB

Para interactuar con MariaDB desde C#, es fundamental crear un usuario que tenga los permisos necesarios para realizar operaciones en la base de datos. La creación de un usuario en MariaDB se realiza mediante comandos SQL. con HeidiSQL al instalarla. A continuación se podrán crear más usuarios con diferentes privilegios (Figura 3.1).

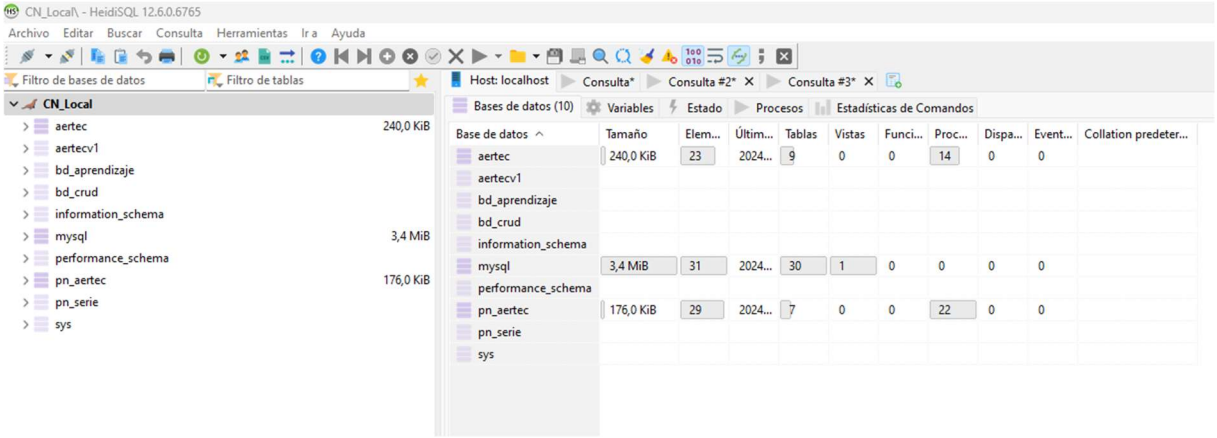


Figura 3.1 *Interfaz Gráfica HeidiSQL. Elaboración propia*

Se puede observar que por defecto ya se instala con ejemplos de bases de datos, esquemas y procedimientos almacenados.

Seguidamente se crea user_pn para trabajar en local, hay que tener en cuenta el puerto configurado que por defecto es el 3306. Después se necesitará esta información para conectar a la base de datos desde Visual Estudio. Véase (Figura 3.2).

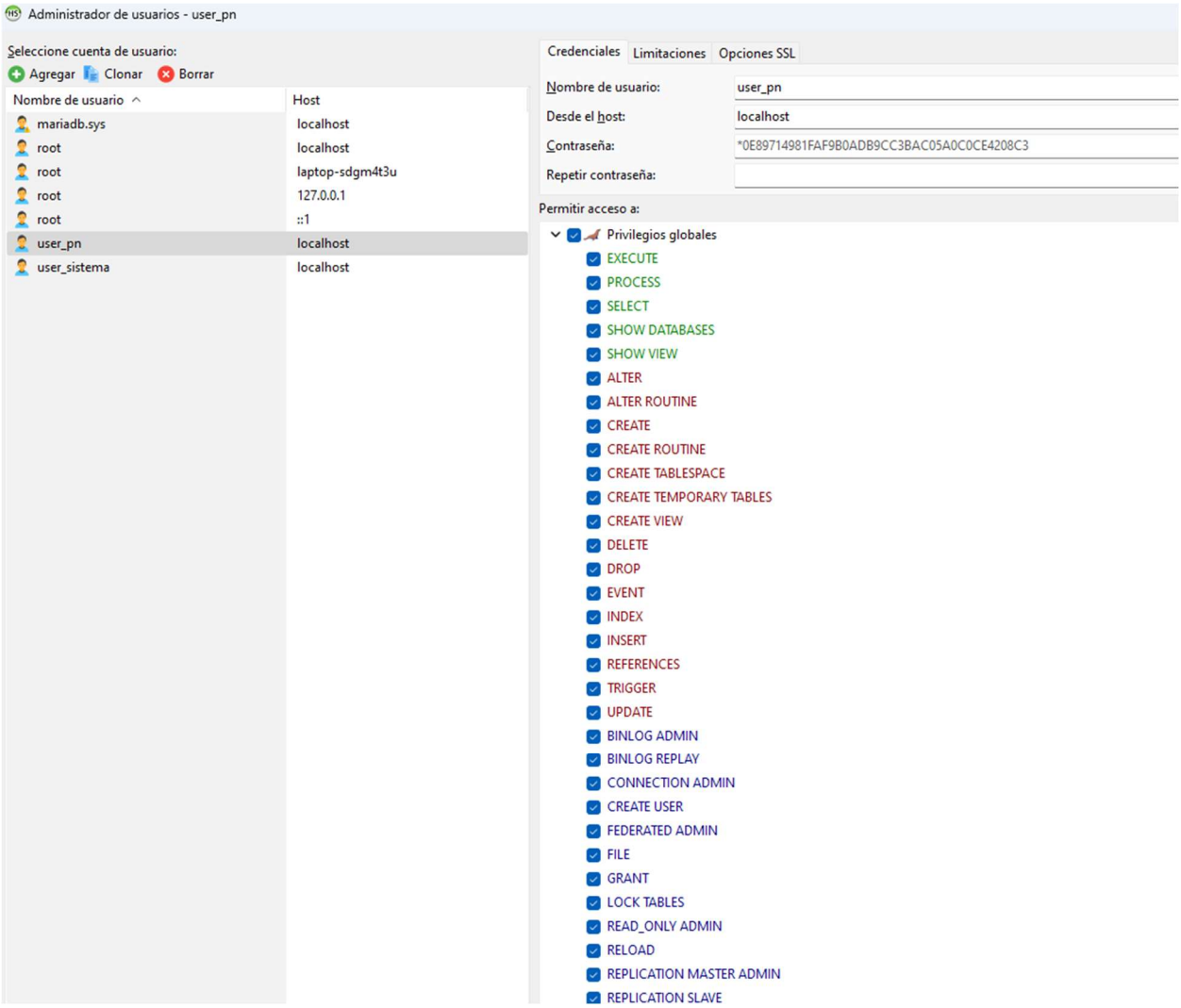


Figura 3.2 *Usuario y Privilegios en MariaDB. Elaboración propia*

3.2. CREACIÓN DE LA BASE DE DATOS Y TABLAS

Una vez dentro de la basa de datos creada, se pueden generar las tablas mediante menú desplegable y a posteriori definir las mismas, o mediante comando SQL en la pestaña consulta.(Figura 3.3).

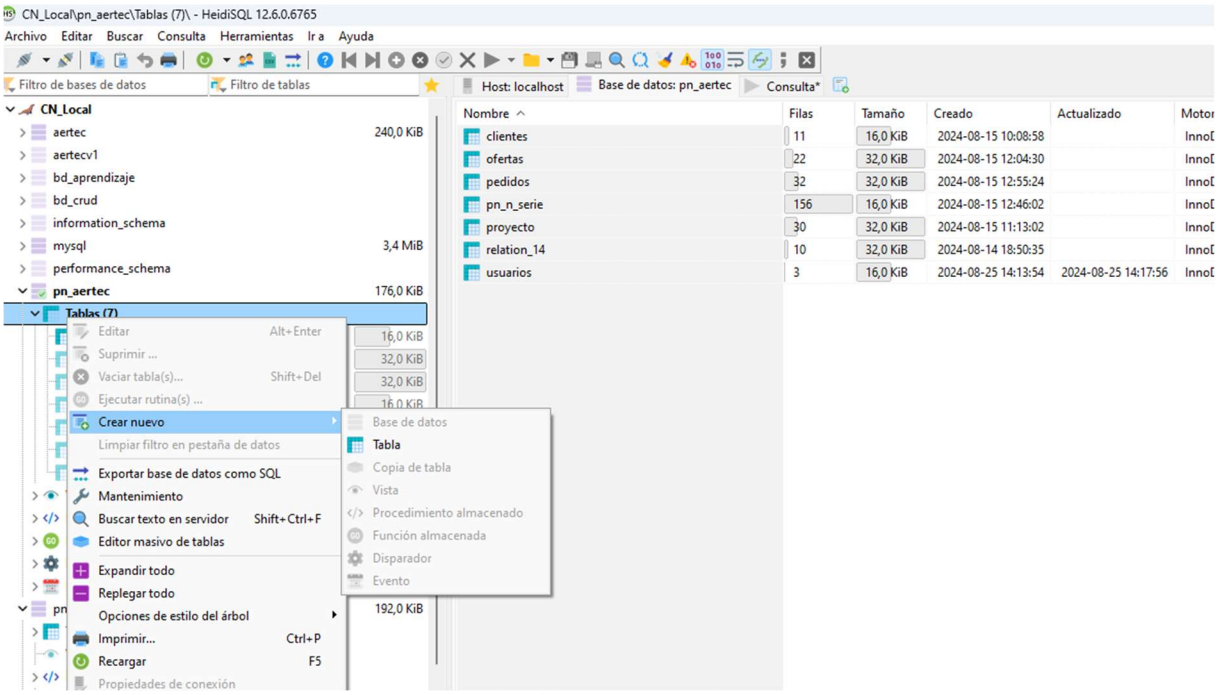


Figura 3.3 *Creación de Tablas en MariaDB. Elaboración propia*

A continuación, se crea la tabla con las diferentes columnas y se definen los tipos de datos, la longitud de estos y las características, también la obligatoriedad y permisos. También el valor Null o sin valor predeterminado,(Figura 3.4)

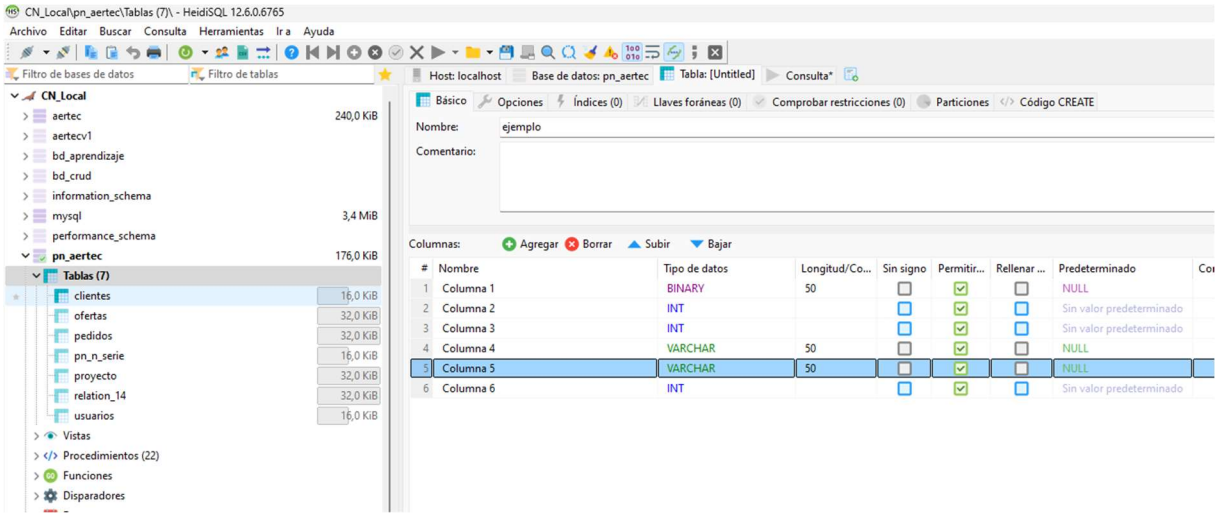


Figura 3.4 *Definición de las Tablas en MariaDB. Elaboración propia*

La otra opción es mediante consulta y código SQL (Figura 3.5). En Anexo 3, se adjunta el código SQL de todas las tablas del proyecto.

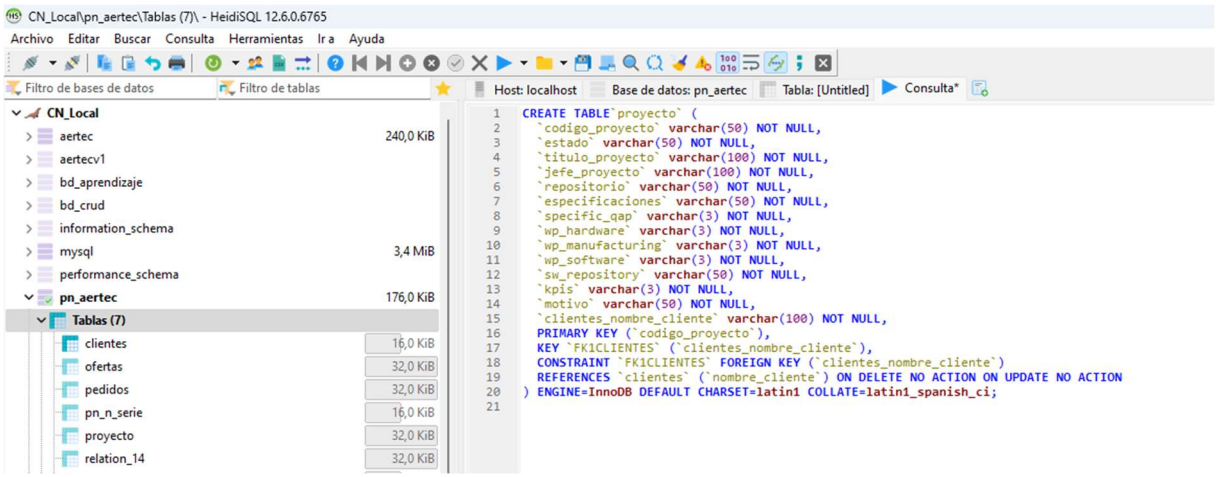


Figura 3.5 *Definición de la Tabla Proyecto con Consulta SQLen MariaDB. Elaboración propia*

3.3. CREACIÓN DE PROCEDIMIENTOS ALMACENADOS EN MARIADB

Los procedimientos almacenados son conjuntos de instrucciones SQL que se almacenan en el servidor de base de datos. Es un programa que se almacena y se ejecuta en el servidor de base de datos. Este programa puede contener una serie de comandos SQL y lógica de programación para realizar operaciones complejas en la base de datos, como cálculos, actualizaciones de registros, inserciones masivas, y mucho más. Los procedimientos almacenados en MariaDB se crean utilizando la instrucción CREATE PROCEDURE y pueden aceptar parámetros de entrada y salida. Estos procedimientos permiten realizar operaciones repetitivas de forma más eficiente y pueden mejorar el rendimiento de las aplicaciones al reducir el tráfico entre la aplicación y la base de datos. Además, proporcionan una capa adicional de seguridad al ocultar la lógica de negocio en la base de datos. Como resumen diremos que sus ventajas son:

- Rendimiento Mejorado: Los procedimientos almacenados pueden mejorar el rendimiento de la aplicación, ya que reducen el tráfico de red al realizar operaciones complejas en el servidor.
- Reutilización de Código: Una vez creados, los procedimientos almacenados pueden ser reutilizados en diferentes aplicaciones y escenarios.
- Seguridad: Al encapsular la lógica de negocio en la base de datos, los procedimientos almacenados protegen la lógica de negocios de la exposición directa a través de la aplicación.
- Mantenimiento Simplificado: Actualizar un procedimiento almacenado es más sencillo y menos propenso a errores que actualizar el código de aplicación.

3.3.1. Creación de un Procedimiento Almacenado en MariaDB

Para ilustrar la creación de un procedimiento almacenado, se considera que desea crear un procedimiento que liste todos los proyectos y otro que permita insertar un nuevo proyecto en la base de datos.

La tabla Proyecto tiene los campos que se visualizan en (Figura 3:6)

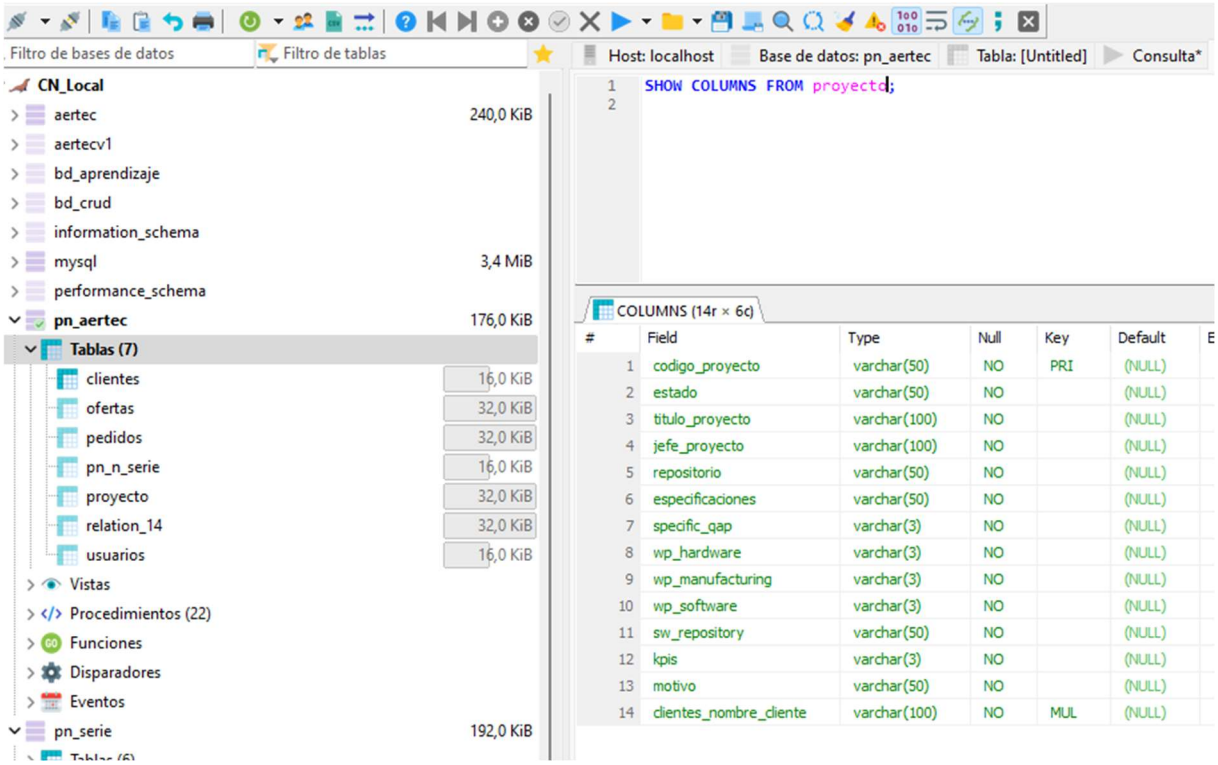


Figura 3.6 *Columnas la Tabla Proyecto con comando SHOW. Elaboración propia*

El siguiente procedimiento almacenado (Figura 3.7); llamado ListarProyectos, seleccionará y devolverá todos los registros de la tabla productos. En el Anexo 4 están los procedimientos almacenados para realizar CRUD de las diferentes tablas.

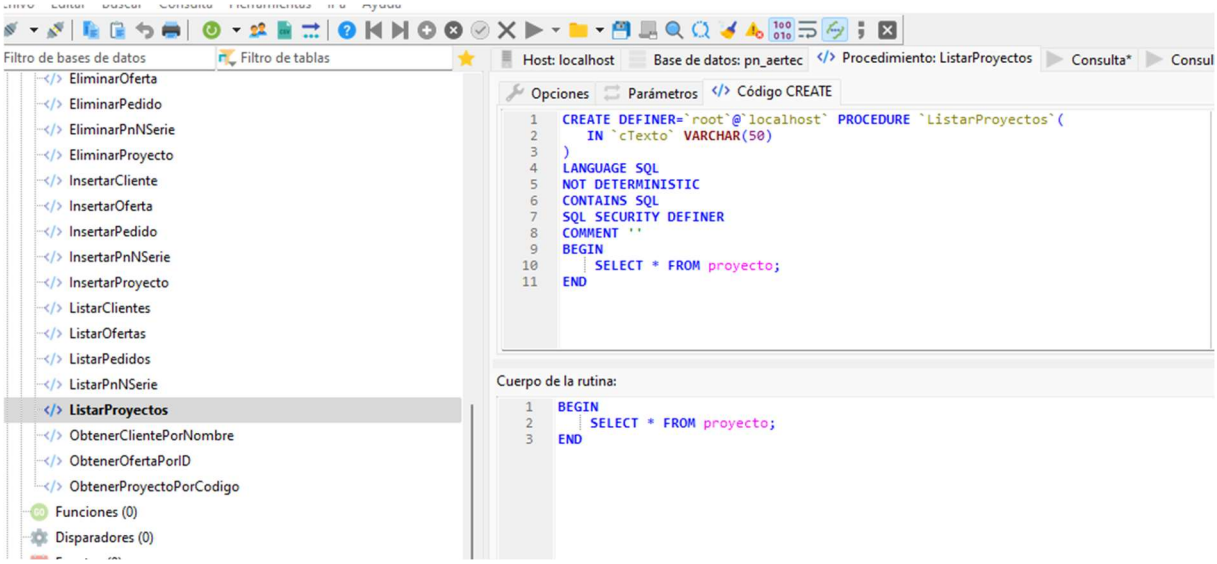


Figura 3.7 *Creación Procedimiento de Listar los Proyectos. Elaboración propia*

En la (Figura 3.8) se puede observar el insertar los valores de un nuevo proyecto ya que esto será de utilidad para el proceso CRUD desde Visual Estudio.

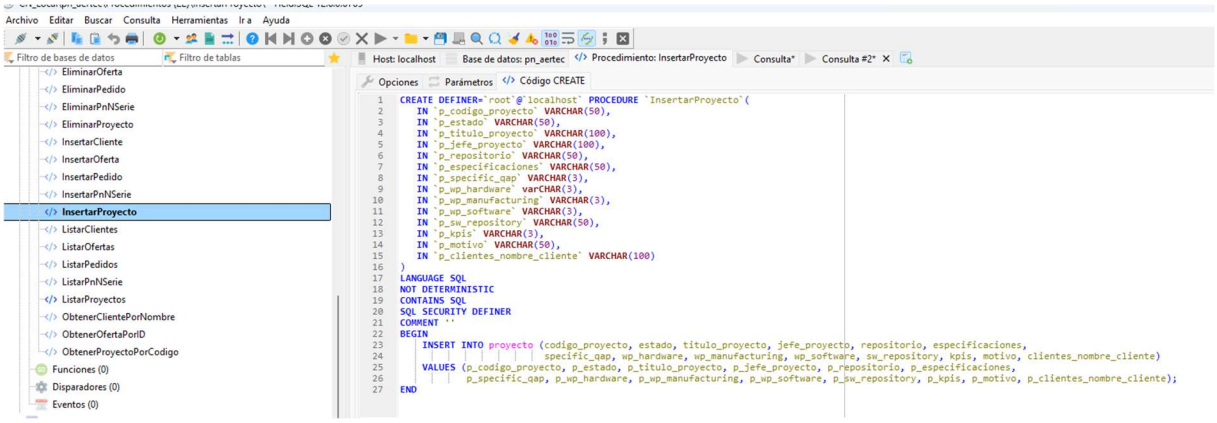


Figura 3.8 *Creación Procedimiento de Insertar Proyectos. Elaboración propia*

Una vez creados, los procedimientos almacenados se pueden llamar desde la línea de comandos de MariaDB o desde cualquier cliente que pueda ejecutar instrucciones SQL. CALL ListarProyectos(); (Figura 3.9). Con estas pruebas nos garantizamos que los procedimientos cumplen con los requisitos con los que se han creado.

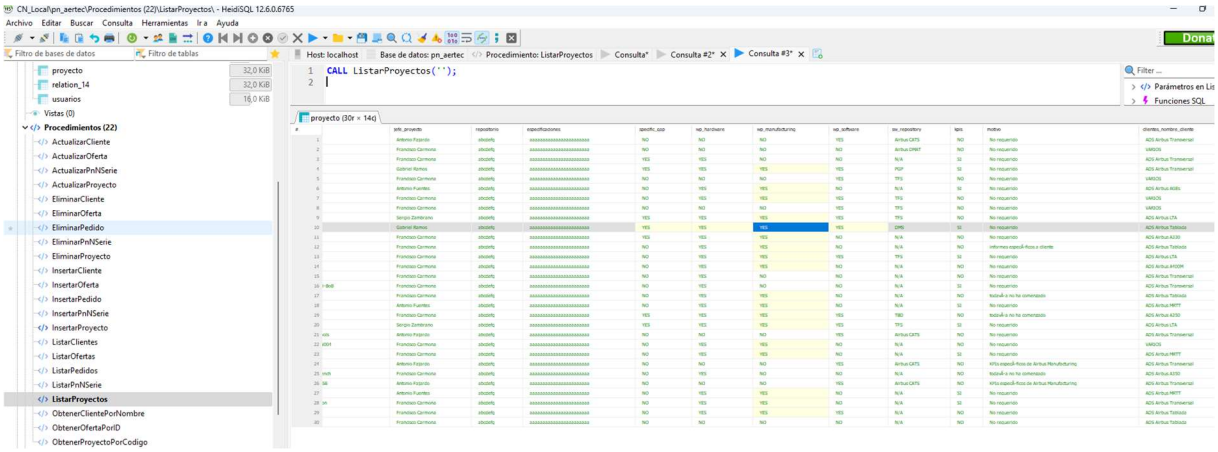


Figura 3.9 *Prueba Procedimiento Listar Proyectos. Elaboración propia*

3.3.2. Crear Procedimiento con JOIN

Los diferentes tipos de JOIN en SQL permiten combinar datos de varias tablas de manera flexible y eficiente, basándose en las relaciones definidas por las claves primarias y foráneas. Entender cómo usar estos JOIN correctamente es fundamental para manipular y consultar datos relacionales en SQL.(Figura 3.10)

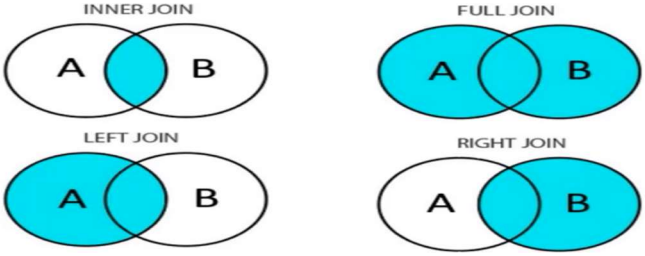


Figura 3.10 *Relación JOIN entre Tablas. Elaboración propia*

En SQL, además de INNER JOIN y LEFT JOIN, existen otros tipos de comandos JOIN que se utilizan para combinar filas de dos o más tablas basadas en una condición relacionada. A continuación, se explica cada uno de ellos:

- INNER JOIN: Combina filas de dos tablas donde existe una coincidencia en ambas tablas. Solo devuelve las filas donde existe una coincidencia en ambas tablas. Comúnmente se utiliza para combinar dos tablas que tienen una relación de clave primaria y clave foránea. La clave primaria de una tabla se empareja con la clave foránea de otra tabla para identificar filas relacionadas.
- LEFT JOIN: Devuelve todas las filas de la tabla de la izquierda, y las filas coincidentes de la tabla de la derecha. Si no hay coincidencia, se devuelve NULL en las columnas de la tabla de la derecha. Se usa cuando se quiere asegurar que todas las filas de la tabla izquierda aparecen en el resultado. Se utiliza cuando se desea obtener todas las filas de una tabla principal (generalmente la que contiene la clave primaria) y las filas relacionadas de otra tabla (clave foránea), incluso si no existe una coincidencia.
- RIGHT JOIN: Devuelve todas las filas de la tabla de la derecha, y las filas coincidentes de la tabla de la izquierda. Si no hay coincidencia, se devuelven NULL en las columnas de la tabla de la izquierda. Es similar al LEFT JOIN, pero con la prioridad invertida; es decir, todas las filas de la tabla de la derecha aparecen en el resultado. Utilizado cuando se desea obtener todas las filas de la tabla de la derecha (clave foránea) y las filas relacionadas de la tabla de la izquierda (clave primaria), incluso si no existe una coincidencia.
- FULL JOIN: Devuelve todas las filas cuando hay una coincidencia en ambas tablas. Las filas que no tienen coincidencias en ambas tablas también se incluirán en el resultado, con NULL en las columnas donde no hay coincidencias. Combina el resultado de LEFT JOIN y RIGHT JOIN. Este tipo de JOIN no es soportado directamente por MySQL o MariaDB, pero puede emularse con una combinación de LEFT JOIN y UNION.
- CROSS JOIN: Devuelve el producto cartesiano de las dos tablas, es decir, cada fila de la primera tabla se combina con cada fila de la segunda tabla. Se usa cuando quieres combinar todas las filas posibles entre dos tablas. No requiere una condición ON.
- SELF JOIN: Es una JOIN de una tabla consigo misma. Se utiliza cuando una tabla tiene una relación jerárquica consigo misma. Normalmente se usa con un alias de tabla para poder referirse a la tabla dos veces en la misma consulta.

En SQL, los comandos JOIN se utilizan para combinar filas de dos o más tablas en función de una condición relacionada. La condición de relación generalmente se basa en claves primarias y foráneas, que son fundamentales para definir la relación entre tablas en una base de datos relacional.

3.4. CREACIÓN DE LA SOLUCIÓN Y PROYECTO EN VISUAL STUDIO

En esta sección se muestra como creará una solución y un proyecto en C# utilizando Visual Studio.

Lenguaje de programación C#; (pronunciado "C Sharp") es un lenguaje de programación diseñado por Microsoft para la infraestructura de lenguaje común. Su sintaxis está basada en C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar a Java. El nombre C Sharp se inspira en el signo musical "#", que se lee como "sharp" en inglés.

Características clave de C#:

- Facilita la comprensión para aquellos familiarizados con estos lenguajes.
- Biblioteca de clases de la plataforma .NET y proporciona una amplia gama de funcionalidades y herramientas para el desarrollo de software.
- Mejoras derivadas de otros lenguajes incorporando mejoras y características de otros lenguajes de programación.
- Amplia documentación y recursos en línea, incluyendo guías, blogs y especificaciones del lenguaje.

3.4.1. Crear una Nueva Solución:

Abrir Visual Studio y seleccionar "Crear un nuevo proyecto". Elegir "Aplicación de Windows Forms" (Figura 3.11)

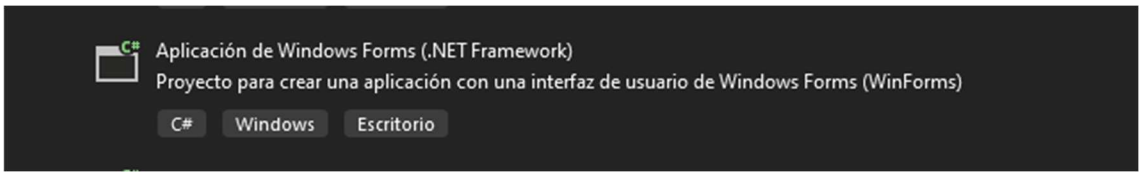


Figura 3.11 *Apertura Proyecto Visual estudio. Elaboración propia*

Configurar el Proyecto como MTD_CRUD_PN y nombrar solución SOL_MTD en el marco .NET Frameword 4.7.2 (Figura 3.12) última versión estable disponible en la red.

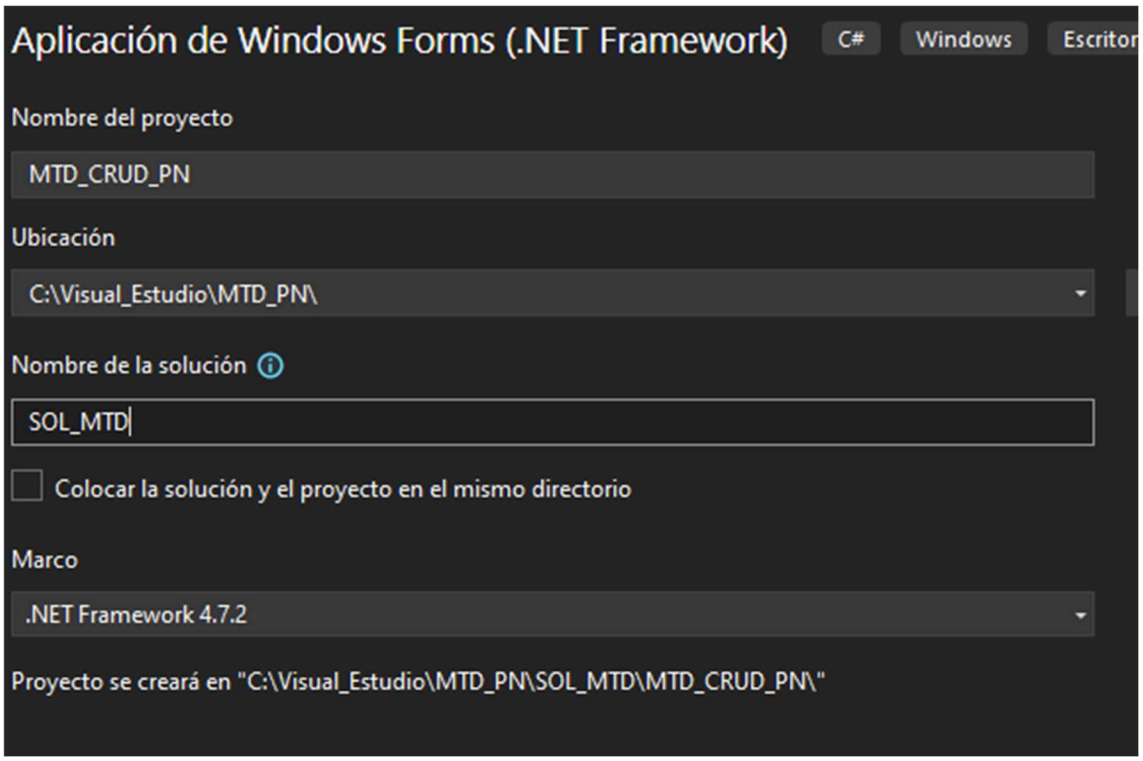


Figura 3.12 *Apertura Proyecto Visual estudio. Elaboración propia*

3.4.2. Instalación del Conector MySQL

Para conectar la aplicación C# con MariaDB, es necesario instalar el conector MySQL para .NET.A continuación se procede a la instalación del MySQL Connector: Abrir el "Administrador

de paquetes NuGet" en Visual Studio y buscar MySql.Data o MySqlConnector. Seguidamente se instala el paquete en el proyecto. Una vez descargado de “https://dev.mysql.com/downloads/connector/net/ “se instala en Archivos de programas. Es un proveedor de datos ADO.NET con soporte para MariaDB Server.

3.4.3. Estructura en Capas

Se construye una arquitectura en capas, con el fin de separar la lógica de acceso a datos del interfaz de usuario. Esto facilita el mantenimiento, la escalabilidad y la reutilización del código. (Figura 3:13) Se utilizan las siguientes capas:

- Capa de Presentación; es responsable de interactuar con el usuario final. Incluye todos los elementos de la interfaz de usuario, como formularios, vistas, controladores. Se maneja la interacción con el usuario con formularios (WinForms), proyecto de aplicación de consola, Windows Forms.
- Capa de Acceso a Datos, es responsable de interactuar con la base de datos o cualquier otro sistema de almacenamiento de datos. Gestiona la lógica para la creación, lectura, actualización y eliminación (CRUD) de datos, acceder y gestionar la persistencia de datos clases de repositorio, métodos de acceso a datos, conexiones a la base de datos, mapear como Entity Framework. Biblioteca de clases en .NET.
- Capa de Entidades; se incluye una capa adicional para definir los modelos de dominio o las entidades de negocio. Esta capa contiene las clases que representan los objetos de dominio y las estructuras de datos que serán utilizadas entre formularios y la capa de acceso a datos.

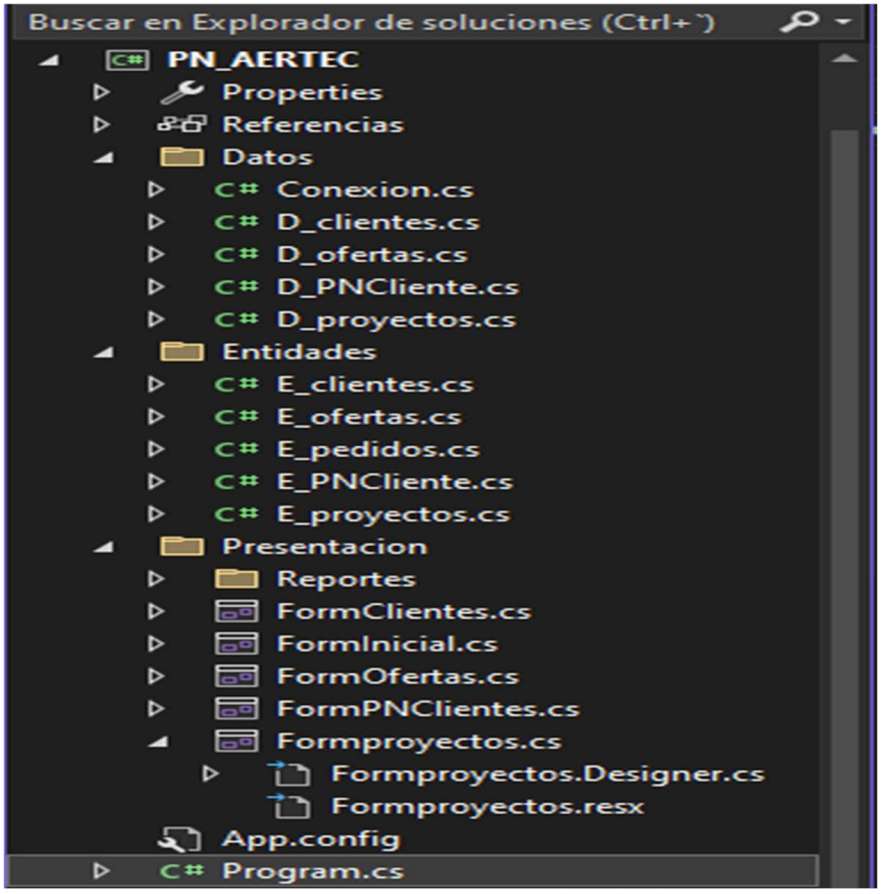


Figura 3.13 *Estructura Carpetas Proyecto Visual estudio. Elaboración propia*

Relaciones entre la capa de presentación interactúa directamente con la capa de entidades. Esta capa nunca debe interactuar directamente con la capa de acceso a datos.

La capa entidades interactúa tanto con la presentación como con datos.

La capa datos interactúa directamente con la base de datos y nunca debe ser accedida directamente desde la capa de presentación.

Cada capa tiene una responsabilidad única, lo que facilita la comprensión y el mantenimiento del código, el cambio en una capa (por ejemplo, cambiar la fuente de datos) no afectará a otras capas, siempre y cuando las interfaces públicas permanezcan iguales. Facilita la prueba unitaria ya que cada capa puede ser probada independientemente de las demás. Permite que diferentes equipos trabajen en diferentes capas simultáneamente.

3.4.4. Creación de la Clase de Conexión

Se debe crear una clase en C# que maneje la conexión a la base de datos MariaDB. Añadir una nueva clase en el proyecto llamada Conexion.cs en la carpeta datos. Se define el método de conexión (Figura 3.14). Para este archivo usaremos los datos de puerto, clave, servidor y nombre de la base de datos, definidos con anterioridad en MariaDB.

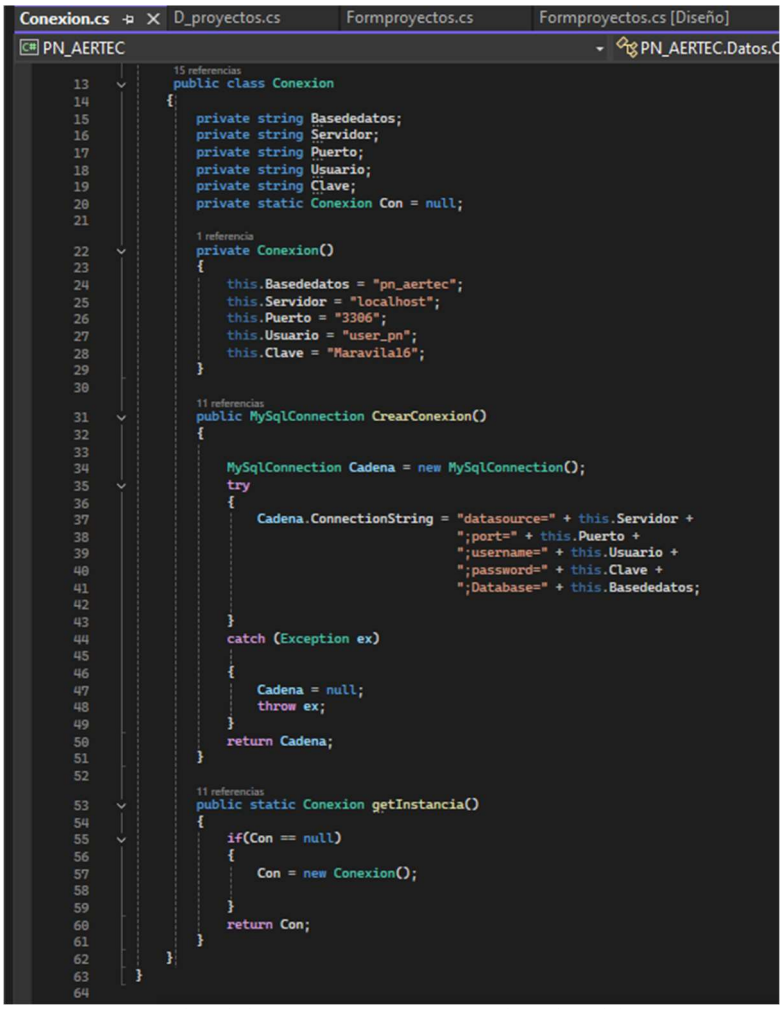


Figura 3.14 *Código de la Conexión MariaDB y Visual estudio. Elaboración propia*

3.4.5. Diseño del Formulario de Mantenimiento

El diseño de un formulario de mantenimiento en la aplicación para gestionar los productos (crear, editar, eliminar, listar) se realiza de la siguiente manera (Figura3.15). Pasos para seguir:

- Crear Formulario: Añadir un nuevo formulario Windows Forms en el proyecto llamado Formproyectos.
- Diseñar la Interfaz de Usuario: Utilizar controles de interfaz de usuario como TextBox, Button, DataGridView (DGV). Para permitir al usuario interactuar con la base de datos. Estos elementos se distribuyen en paneles, uno donde se debe rellenar datos de la tabla; panel central de trabajo y a la izquierda panel de botones, opciones principales. En la parte inferior el DGV donde se listan los contenidos de las tablas completas o según la búsqueda que se demande.

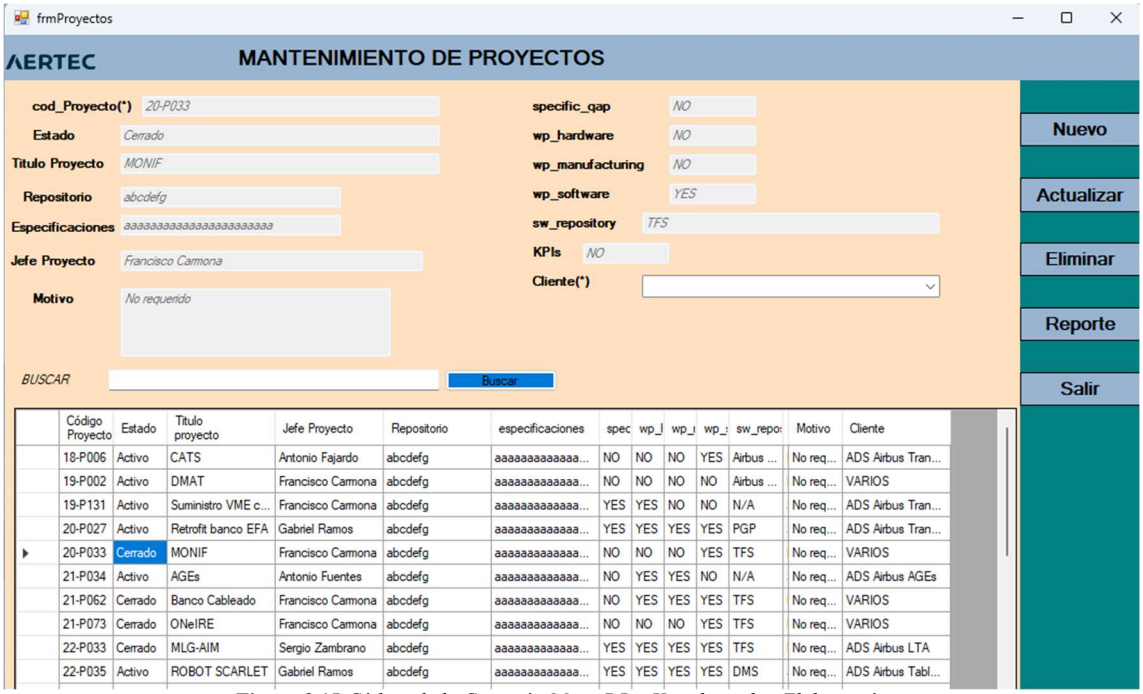


Figura 3.15 *Código de la Conexión MariaDB y Visual estudio. Elaboración propia*

Es en este paso donde se define mediante regiones y métodos, tanto las variables, los botones y acciones que deben realizar, como el estado de estos. Para más detalle véase ejemplo en el Anexo 5, de uno de los formularios (Proyectos).

Al mismo tiempo se debe completar en el archivo de datos D_Proyectos las correspondientes acciones, y comprobar que funcionan correctamente antes de avanzar en cada funcionalidad del proceso CRUD.

3.4.6. Crear Entidad E_proyectos.

Es necesario crear una clase que represente la entidad de proyectos para gestionar los datos de manera más estructurada. La entidad encapsula toda la información relevante sobre los proyectos que la empresa maneja. Esto no solo mejora la seguridad al proteger los datos de modificaciones indebidas, sino que también asegura que cualquier cambio en los atributos del

proyecto sea realizado a través de métodos controlados, donde se pueden validar. También facilita la integración con otras funcionalidades, como la generación de informes, donde los métodos de la clase pueden ser utilizados para filtrar y presentar datos específicos. Este enfoque permite que la aplicación sea más robusta, segura y fácil de mantener, proporcionando una base sólida para la gestión de datos de proyectos dentro de la organización. (Figura 3.16)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PN_AERTEC.Entidades
{
    3 referencias
    public class E_proyectos
    {
        1 referencia
        public int ID_proyecto { get; set; }
        2 referencias
        public string Codigo_proyecto { get; set; }
        2 referencias
        public string Estado { get; set; }
        2 referencias
        public string Titulo_proyecto { get; set; }
        1 referencia
        public string Repositorio { get; set; }
        2 referencias
        public string Jefe_Proyecto { get; set; }
        1 referencia
        public string especificaciones {  get; set; }
        2 referencias
        public string specific_qap { get; set; }
        2 referencias
        public string wp_hardware { get; set; }
        2 referencias
        public string wp_manufacturing { get; set; }
        2 referencias
        public string wp_software { get; set; }
        2 referencias
        public string sw_repository { get; set; }
        2 referencias
        public string KPIs { get; set; }
        2 referencias
        public string Motivo { get; set; }

    }
}
```

Figura 3.16 *Código E_proyectos en Visual estudio. Elaboración propia*

3.4.7. Crear Resto de Formularios para el Mantenimiento

En la clase D_proyectos que maneje todas las operaciones de la base de datos relacionadas con la entidad véase (Anexo 6). Se implementan métodos CRUD:

- Código para Crear.
- Código para Actualizar.
- Código para Leer.
- Código para Borrar.

Este proceso debe realizarse con todas las tablas de la base de datos, tanto en los formularios en entidades y como en datos.

4. FORMULARIO DE ACCESO Y CIBERSEGURIDAD

Este capítulo detalla los formularios de acceso creados para la aplicación, que permiten a los usuarios autenticarse y acceder a las funcionalidades del sistema según sus roles. Se ha implementado un sistema de autenticación que verifica las credenciales de los usuarios (nombre de usuario y contraseña) al momento de ingresar a la aplicación. Dependiendo del rol asignado a cada usuario (como administrador o usuario estándar), se habilitan o restringen ciertas funcionalidades, asegurando que solo los usuarios autorizados tengan acceso a funciones críticas del sistema. Este enfoque mejora la seguridad del sistema y permite una gestión más eficiente y organizada de los permisos y privilegios dentro de la aplicación.

4.1 FORMULARIOS DE ACCESO Y AUTENTICACIÓN EN LA APLICACIÓN

En esta sección, se describen los formularios de acceso diseñados para la aplicación, los cuales son esenciales para la autenticación de los usuarios y la gestión de permisos. El objetivo principal de estos formularios es garantizar que solo los usuarios autorizados puedan acceder a las diferentes funcionalidades del sistema, dependiendo de sus roles asignados. Esta autenticación robusta mejora la seguridad de los datos y la integridad del sistema.

4.1.1. Diseño del Formulario de Acceso

El formulario de acceso es la primera interfaz con la que interactúan los usuarios al iniciar la aplicación. Este formulario solicita las credenciales del usuario, es decir, el nombre de usuario y la contraseña (Figura 4.1). La interfaz del formulario está diseñada para ser intuitiva y fácil de usar, con campos claramente etiquetados y mensajes de error específicos que guían al usuario en caso de ingreso de datos incorrectos código en (Anexo 7).

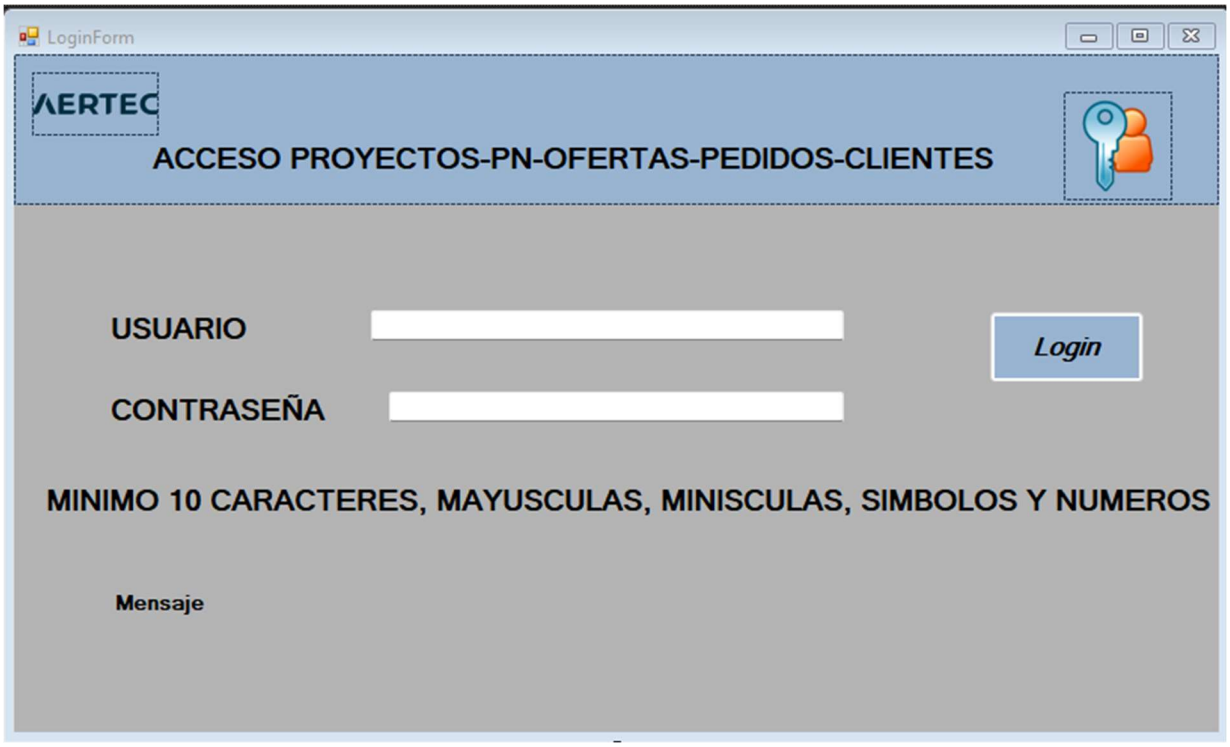


Figura 4.1 *Formulario de acceso a la aplicación. Elaboración propia*

Elementos del Formulario de Acceso:

- Campo de Nombre de Usuario: Un campo de texto donde el usuario debe ingresar su nombre de usuario registrado.
- Campo de Contraseña: Un campo de contraseña que oculta los caracteres mientras el usuario ingresa su clave, proporcionando una capa adicional de seguridad.
- Botón de Ingreso: Un botón que, al ser presionado, activa el proceso de validación de credenciales.
- Botón de Cancelación: Permite al usuario cerrar el formulario de acceso o limpiar los campos ingresados.
- Mensaje de Error: Un área donde se muestran mensajes de error si las credenciales son incorrectas o si el usuario intenta acceder sin llenar todos los campos necesarios.

4.1.2. Proceso de Autenticación

El proceso de autenticación se inicia cuando el usuario ingresa sus credenciales y hace clic en el botón de ingreso. A continuación, se detallan los pasos que sigue la aplicación para autenticar a un usuario:

- Recopilación de Credenciales: El formulario recoge los datos introducidos en los campos de nombre de usuario y contraseña.
- Validación de Datos en el Lado del Cliente: Antes de enviar los datos al servidor, se realiza una validación básica en el cliente para asegurar que ambos campos no estén vacíos. Si los campos están vacíos, se muestra un mensaje de error al usuario.
- Conexión a la Base de Datos: Si los datos son válidos, se establece una conexión con la base de datos MariaDB utilizando las credenciales y la cadena de conexión definida en la aplicación. Código (Anexo 8)
- Consulta de Validación: Se ejecuta una consulta SQL en la base de datos para verificar si existe un registro coincidente con el nombre de usuario y la contraseña proporcionados. Esta consulta generalmente utiliza una declaración SQL SELECT código en (Anexo 9) con una cláusula WHERE que compara las credenciales ingresadas con las almacenadas en la tabla de usuarios Figura 4-2

#	Nombre	Tipo de datos	Longitud/Co...	Sin signo	Permitir.
1	id_usuario	INT	11	☐	☐
2	nombre_usuario	VARCHAR	50	☐	☐
3	clave	VARCHAR	100	☐	☐
4	rol	ENUM	'admin','usu...	☐	☑
5	activo	TINYINT	1	☐	☑

Figura 4.2 *Tabla usuarios en MariaDB. Elaboración propia*

- Verificación de Resultados: Si la consulta devuelve un resultado, se considera que las credenciales son correctas, y se procede a la siguiente fase del proceso de autenticación. Si no se encuentran coincidencias, se muestra un mensaje de error al usuario indicando que las credenciales son incorrectas. Procedimiento almacenado “verificar_usuario” en mariaDB.
- Asignación de Roles y Permisos: Una vez autenticado, el sistema verifica el rol del usuario (por ejemplo, administrador, usuario estándar, etc.) y ajusta dinámicamente la interfaz de

usuario para habilitar o restringir ciertas funciones según los permisos asociados con el rol del usuario. Este proceso se maneja mediante un control de flujo de trabajo que determina qué funcionalidades están disponibles para el usuario actual.

- Acceso a la Aplicación: Si la autenticación es exitosa, el usuario es redirigido a la página principal de la aplicación o al panel de control correspondiente a su rol. En este punto, el usuario puede comenzar a interactuar con la aplicación de acuerdo con sus permisos.

4.1.3. Seguridad en la Autenticación

Para proteger la información de usuario y la integridad del sistema, se deberá implementar varias medidas de seguridad en el proceso de autenticación de acuerdo con las normativas internas de la compañía:

- Hashing de Contraseñas: Las contraseñas no se almacenan en texto plano en la base de datos. En cambio, se utilizan técnicas de hashing para convertir la contraseña en una cadena única de caracteres que no puede ser revertida a su forma original fácilmente. Esto protege las contraseñas en caso de que la base de datos sea comprometida.
- Consulta Parametrizada: Las consultas SQL para la autenticación deberán estar parametrizadas para prevenir ataques de inyección SQL. En lugar de concatenar directamente los valores de las credenciales en la consulta, se utilizan parámetros SQL para manejar los datos del usuario.
- Manejo de Sesiones: Una vez que un usuario en red corporativa sea autenticado, se deberá crear una sesión única para ese usuario. Esta sesión se utiliza para rastrear la actividad del usuario y verificar la autenticación en cada solicitud posterior.
- Bloqueo de Cuentas: Se deberá implementar una política de bloqueo de cuentas después de varios intentos fallidos de autenticación para prevenir ataques de fuerza bruta.

4.1.4. Gestión de Roles y Permisos

El sistema está diseñado para manejar múltiples roles de usuario, cada uno con permisos específicos que determinan qué funcionalidades están disponibles. Esto asegura que los usuarios solo tengan acceso a las partes de la aplicación que necesitan para realizar sus tareas. Este apartado no se ha completado ya que depende de aprobaciones de varios departamentos, por lo que solo se ha utilizado dos perfiles.

Roles de Usuario Comunes:

- Administrador: Acceso completo a todas las funcionalidades del sistema, incluyendo la gestión de usuarios y la configuración del sistema.
- Usuario Estándar: Acceso limitado a funcionalidades específicas, como la visualización y edición de registros de proyectos.

4.2. MEJORAS FUTURAS Y CONSIDERACIONES

Para mejorar la seguridad y funcionalidad del sistema de autenticación en el futuro, se pueden considerar las siguientes mejoras:

- Autenticación Multifactor (MFA): Implementar un segundo factor de autenticación (como un código enviado a un dispositivo móvil) para aumentar la seguridad.

- Integración con LDAP: Permitir la autenticación de usuarios utilizando servicios de autenticación externa como LDAP (Lightweight Directory Access Protocol) o OAuth para un manejo más eficiente de usuarios en entornos corporativos.
- Monitoreo de Accesos y Auditoría: Implementar un sistema de monitoreo y auditoría que registre los intentos de acceso y cualquier actividad sospechosa para identificar y responder a posibles amenazas de seguridad.
- Uso de Protocolos Seguros: Asegurar la utilización de protocolos seguros como HTTPS para proteger los datos transmitidos entre el cliente y el servidor. Esto garantiza que los datos no se puedan interceptar ni manipular durante la transmisión.
- Cifrado de Datos Sensibles: Implementar el cifrado de datos sensibles, tanto en tránsito como en reposo. Utilizar algoritmos de cifrado fuertes (como AES-256) para proteger datos críticos almacenados en la base de datos, como contraseñas, información personal o detalles financieros.
- Manejo de Sesiones Seguras: Implementar tokens de sesión únicos y asegurar que se invaliden adecuadamente cuando el usuario cierra sesión o después de un tiempo de inactividad.
- Política de Contraseñas Fuertes: Requerir que los usuarios creen contraseñas fuertes que incluyan una combinación de letras mayúsculas y minúsculas, números y caracteres especiales. Considera la implementación de políticas que requieran la renovación periódica de contraseñas y limiten la reutilización de contraseñas anteriores.
- Control de Acceso Basado en Roles (RBAC): Implementar controles de acceso basados en roles para asegurar que los usuarios solo tengan acceso a las funciones y datos que necesitan. Establecer roles y permisos claramente definidos y revocar accesos cuando un usuario ya no necesita o deba tener privilegios específicos.
- Validación de Entradas del Usuario: Proteger la aplicación de ataques de inyección SQL y otros tipos de ataques de inyección mediante la validación estricta de entradas del usuario y el uso de consultas parametrizadas o procedimientos almacenados.

4.3. MEJORES PRÁCTICAS EN LA GESTIÓN DE DATOS Y CONEXIONES A LA BASE DE DATOS.

A continuación, se adjuntan algunas recomendaciones y buenas prácticas para fortalecer la seguridad de la aplicación y mejorar su rendimiento a medida que crece:

- Uso de Conexiones de Base de Datos Seguras: Configurar la base de datos para que solo acepte conexiones desde fuentes de confianza y utiliza firewalls y listas de control de acceso (ACLs) para limitar el acceso a la base de datos. Desactivar las conexiones directas de la base de datos desde clientes no seguros.
- Uso de Procedimientos Almacenados: Utilizar procedimientos almacenados para realizar operaciones de base de datos complejas. Los procedimientos almacenados pueden ayudar a encapsular la lógica de negocio, mejorar el rendimiento y proteger contra ataques de inyección SQL.
- Gestión de Conexiones de Base de Datos: Implementar un manejo adecuado de conexiones de base de datos utilizando un pool de conexiones para mejorar el rendimiento y la eficiencia de la aplicación. Asegurar de cerrar y liberar correctamente las conexiones después de su uso para evitar fugas de memoria y problemas de rendimiento.

- Auditoría y Monitoreo de Actividades: Implementar mecanismos de auditoría y monitoreo para rastrear las actividades en la base de datos y detectar posibles intentos de acceso no autorizado. Configurar alertas para actividades sospechosas y realizar auditorías periódicas de la base de datos y la infraestructura.
- Respaldos y Recuperación Ante Desastres: Implementar estrategias de respaldo y recuperación ante desastres para asegurar que los datos se puedan recuperar en caso de pérdida de datos, fallos en el sistema o ataques de ransomware. Realizar respaldos periódicos y almacenarlos en ubicaciones seguras y fuera del sitio principal.

4.3.1. Optimización de Consultas

Una base de datos bien optimizada es fundamental para el rendimiento general de la aplicación. Aquí se detallan algunas técnicas para optimizar consultas SQL y mejorar el rendimiento de la base de datos:

- Evitar el Uso de SELECT *: Seleccionar únicamente las columnas necesarias en lugar de utilizar SELECT * en las consultas SQL. Esto reduce la cantidad de datos transferidos y mejora el rendimiento de la consulta.
- Uso de Joins Adecuados: Asegurar de usar los tipos de JOIN adecuados (INNER JOIN, LEFT JOIN, etc.) según las necesidades de la consulta. Los JOINs mal utilizados pueden resultar en una gran cantidad de datos no necesarios que pueden ralentizar el rendimiento de la consulta.
- Filtrado y Paginación: Utilizar cláusulas WHERE para filtrar los datos de manera eficiente y evitar el procesamiento de registros innecesarios. Implementar la paginación para trabajar con grandes conjuntos de resultados.
- Optimización de Subconsultas: Evitar el uso de subconsultas complejas. En su lugar, utiliza JOINs para mejorar la legibilidad y el rendimiento de las consultas.
- Evitar el Uso de Funciones en Cláusulas WHERE: Evitar usar funciones en las cláusulas WHERE. Esto puede impedir el uso de índices y ralentizar el rendimiento de la consulta.

4.3.2. Uso Eficiente de Índices y Claves Primarias/Foráneas

En una base de datos es fundamental la gestión de claves primarias y foráneas, por lo que se recomienda para futuras modificaciones o ineficacias de la base de datos:

- Creación de Índices en Columnas de Búsqueda Frecuente: Utilizar índices en columnas que se utilizan frecuentemente en cláusulas WHERE, JOIN o ORDER BY. Los índices pueden acelerar significativamente el rendimiento de las consultas al reducir la cantidad de datos que se necesita buscar.
- Mantenimiento de Índices: Realizar mantenimiento regular de índices para asegurar que no se fragmenten y sigan siendo eficientes. En bases de datos con gran cantidad de inserciones, actualizaciones o eliminaciones, los índices pueden desfragmentarse y perder eficacia.
- Uso de Claves Primarias y Claves Únicas: Definir claves primarias en cada tabla para asegurar la unicidad de registros y mejorar el rendimiento de las consultas. También considera el uso de claves únicas en columnas que deben contener valores únicos (por ejemplo, correos electrónicos o números de identificación).

- Minimización del Uso de Índices en Columnas con Alta Cardinalidad: Evitar usar índices en columnas con alta cardinalidad (muchos valores distintos). Esto puede aumentar el tamaño del índice y reducir su eficiencia.
- Optimización del Uso de Índices Compuestos: Cuando sea apropiado, utilizar índices compuestos en lugar de varios índices individuales para mejorar el rendimiento de consultas que filtran u ordenan por múltiples columnas.
- Eliminación de Índices Innecesarios: Revisar periódicamente los índices que no se utilizan en la ejecución de consultas y eliminarlos. Los índices innecesarios pueden ralentizar las operaciones de escritura (inserciones, actualizaciones, eliminaciones) y aumentar el espacio de almacenamiento requerido.

En resumen, una gestión eficiente de índices y claves en la base de datos no solo mejora el rendimiento de las consultas, sino que también optimiza el uso de recursos y garantiza la integridad y unicidad de los datos. Es esencial realizar una revisión periódica y un mantenimiento adecuado de estos elementos para asegurar una operación óptima y sostenible del sistema.

5. DOCUMENTACION GRAFICA.

En este capítulo se proporcionan varios esquemas de cómo se relacionan y estructuran los datos en la aplicación. Para entender la organización, su integridad y las posibles relaciones entre las diferentes entidades, facilitando el desarrollo y mantenimiento de la aplicación (Anexo 10) presentación realizada al personal que debe utilizar la aplicación; funcionamiento de los diferentes formularios.

5.1. DIAGRAMA DE FLUJO DE LA APLICACIÓN

A continuación, se describen los procesos principales que forman parte del sistema de gestión de ofertas, pedidos, proyectos, y seguimiento de piezas en la base de datos (Figura 5.3). Cada proceso se detalla para resaltar su función y los pasos necesarios para asegurar una gestión eficiente y precisa dentro del entorno de la empresa:

- Alta de Oferta: Este proceso permite al usuario registrar una nueva oferta en la base de datos. La oferta incluirá información como el número de oferta, la fecha, la descripción, el estado, y el cliente asociado.
- Alta de Pedido: Una vez creada una oferta, el usuario puede generar pedidos asociados a esa oferta. Cada pedido contendrá detalles como el número de pedido, la fecha de pedido, la descripción, y el proyecto asociado. Véase (Figuras 5.1 y 5.2). Donde plasman el número de PN del producto y los requerimientos que deben pasar a formar parte de la base de datos.
- Alta de Proyecto: Los pedidos pueden estar relacionados con proyectos nuevos o existentes. Este proceso permite registrar un nuevo proyecto con información relevante como el título del proyecto, la descripción, las fechas importantes, y el jefe de proyecto.
- Seguimiento de Piezas y Componentes (PN): Este proceso asegura que se pueda hacer un seguimiento completo de cada pieza o componente, desde su fabricación hasta su entrega final, utilizando números de pieza únicos (PN) y manteniendo un historial de movimientos.

HERRAMIENTA Nº: PROO-01-2350-00510-001-A (por confirmar)	PROGRAMA: LTA
DESIGNACIÓN: Kit de auriculares de intercomunicación	ESTACIÓN: EDV
FECHA DE SOLICITUD: 14/06/2023	FECHA DE NECESIDAD: LO ANTES POSIBLE

Figura 5.1 *Detalle Caratula Pedido. Elaboración propia*

Requerido 01 El proveedor deberá suministrar 3 auriculares David Clark modelo H10-60H (PN AIRBUS PROO01235000501A).

Requerido 02 El proveedor deberá proporcionar un embalaje de dimensiones adecuadas. Debe ser de PEHD con forma silueteada de cada uno. elemento. No se permite espuma precortada.

Requerido 03 El equipo deberá estar rotulado según NT-MTIM-TIM-430-13005: "Requisitos Técnicos Generales para Etiquetado y Envasado de Medios de Ensayo", indicando el nombre y PN:

EQUIPO

NOMBRE: KIT CASCOS INTERFONIA (Por confirmar)
PN: PROO-01-2350-00510-001-A (por confirmar)

AURICULARES

NOMBRE: CASCOS H10-60H (por confirmar)
PN: PROO-01-2350-00501-007-A (por confirmar)

NOMBRE: CASCOS H10-60H (por confirmar)
PN: PROO-01-2350-00501-008-A (por confirmar)

NOMBRE: CASCOS H10-60H (por confirmar)
PN: PROO-01-2350-00501-009-A (por confirmar)

Requerido 04 El proveedor deberá enviar todos los documentos de acuerdo al párrafo "DOCUMENTOS A PRESENTAR POR EL PROVEEDOR".

Figura 5.2 *Detalle Pedido y Requerimientos. Elaboración propia*

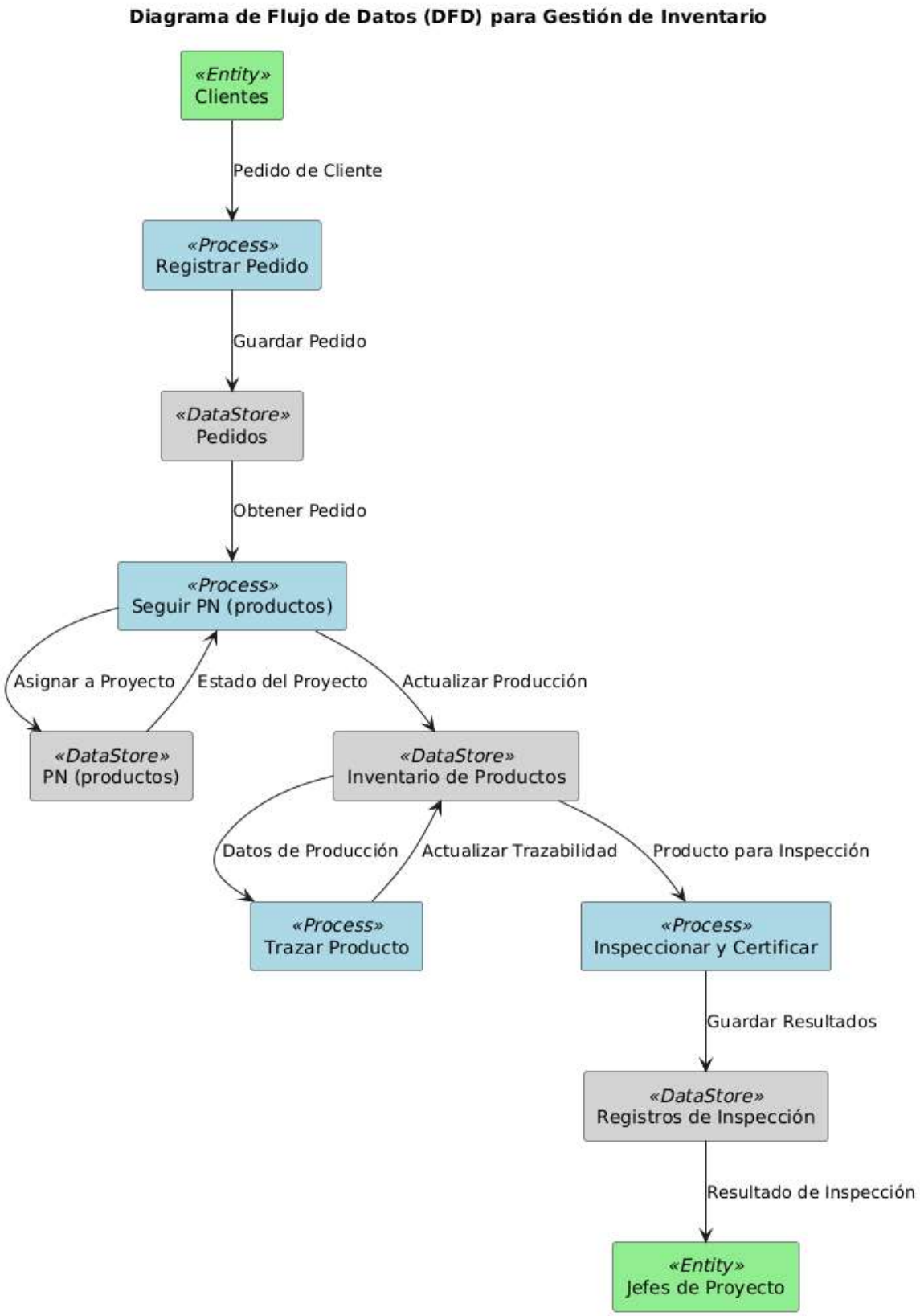


Figura 5.3 *Diagrama de procesos de los PN. Elaboración propia*

5.2. DIAGRAMA DE UN PROCESO CRUD

Este diagrama muestra cómo un usuario interactúa con una aplicación para realizar operaciones de creación, lectura, actualización y eliminación en una base de datos (Figura 5.3). Seguidamente se explica el diagrama:

- Inicio del Diagrama: Usuario iniciando sesión en la aplicación.
- Validación de Credenciales: Verificar si las credenciales ingresadas son correctas. Si no lo son, se muestra un mensaje de error y el proceso termina. Si son correctas abre formulario de inicio. Elegir el formulario (Clientes, Pedidos; Ofertas, Proyectos; PN) sobre el que se quiere realizar el proceso CRUD.
- Selección de Operación CRUD: El usuario selecciona una operación: Crear, Leer, Actualizar o Eliminar.
- Proceso para Cada Operación:

✓ Crear: Muestra un formulario, el usuario ingresa datos, estos son validados, y luego se insertan en la base de datos.
✓ Leer: Solicita datos de la base de datos y los muestra al usuario.
✓ Actualizar: Muestra un formulario con los datos actuales, el usuario realiza modificaciones, se validan los datos, y luego se actualizan en la base de datos.
✓ Eliminar: Solicita confirmación de eliminación. Si se confirma, se elimina el registro; de lo contrario, se cancela la operación.

- Cierre de Sesión: El usuario cierra sesión y el proceso finaliza.

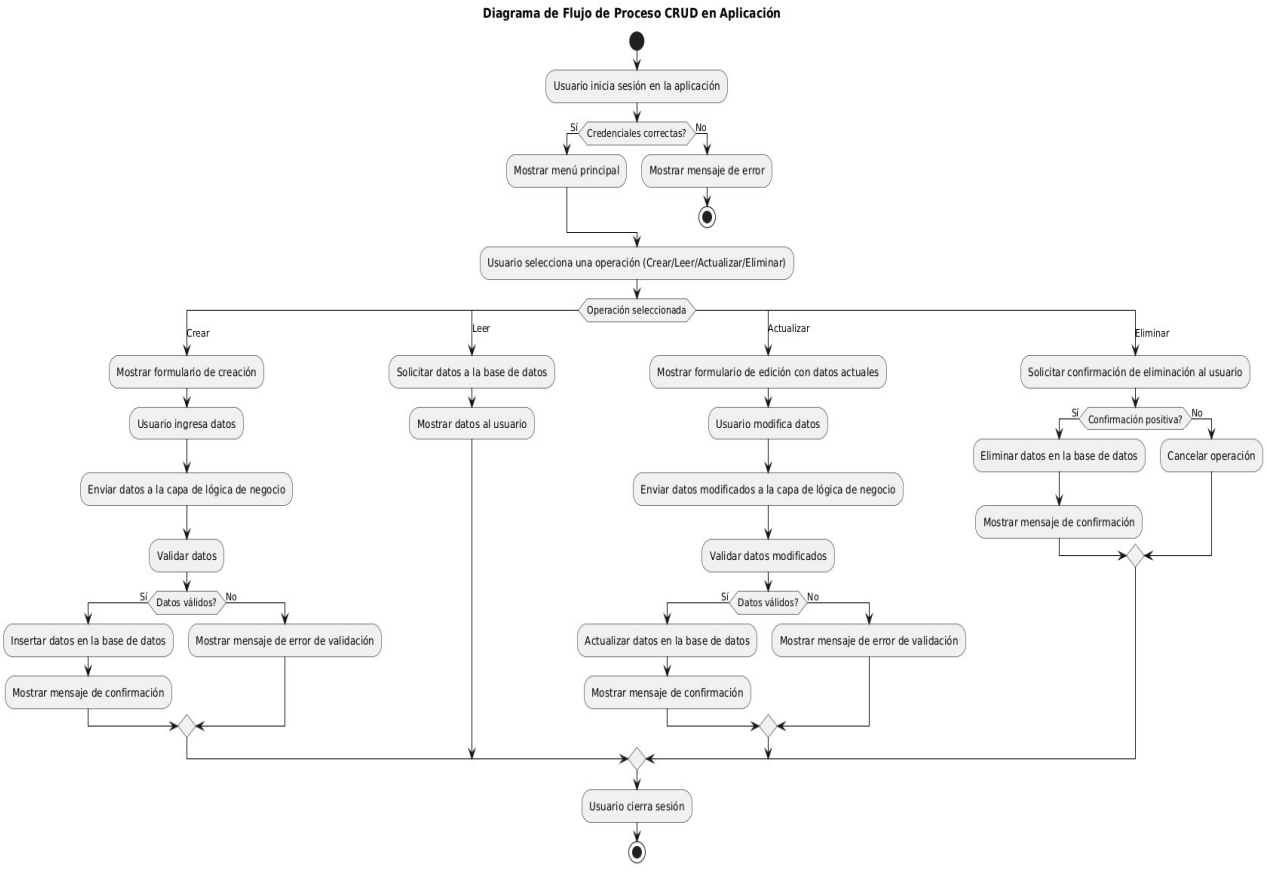


Figura 5.4. *Diagrama Proceso CRUD. Elaboración propia*

5.3. DIAGRAMA DE ARQUITECTURAS EN CAPAS

Este diagrama desglosa la arquitectura del proyecto en diferentes capas. A continuación, se describe cada componente y cómo se interconectan (Figura 5.4):

Componentes de la Capa de Presentación:

- Capa de Presentación:
  - ✓ Formulario de inicio de sesión donde los usuarios ingresan sus credenciales.
  - ✓ El formulario principal que permite la navegación a otras funciones como la administración de proyectos y reportes.
- Componentes de la Capa de Lógica de Negocio:
  - ✓ Servicio para gestionar la autenticación de usuarios.
  - ✓ Servicio para gestionar las operaciones CRUD relacionadas con proyectos.
- Componentes de la Capa de Acceso a Datos:
  - ✓ Repositorio que interactúa con la base de datos para operaciones relacionadas con los usuarios.
  - ✓ Repositorio que maneja las operaciones CRUD para la tabla de proyectos.
- Base de Datos: MariaDB: Base de datos que almacena las tablas necesarias (Usuarios, PN, Proyectos; etc) procedimientos almacenados y vistas.

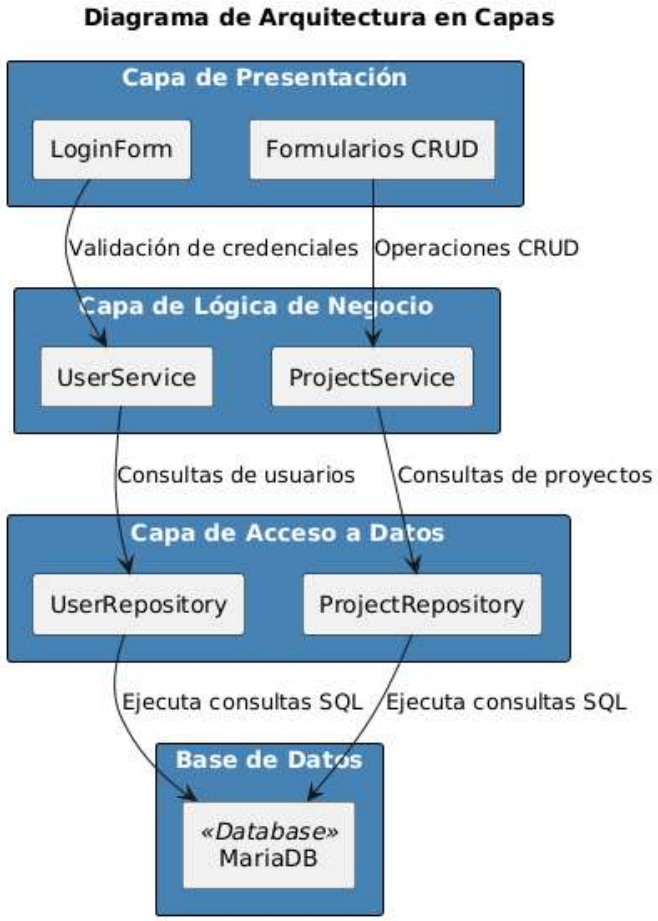


Figura 5.5 *Diagrama Arquitectura por Capas. Elaboración propia*

5.4. DIAGRAMA DE REPORTES

Este diagrama de flujo proporciona una visión clara del proceso reportes, incluyendo la interacción con el DGV, la funcionalidad de buscar, y las opciones de exportación. Se puede personalizar y ampliar este diagrama según las necesidades (Figura 5.6):

- Inicio del Proceso: El usuario se encuentra el DGV para visualizar los datos. Si el DGV está vacío, se muestra un mensaje indicando que no hay datos disponibles y el proceso finaliza. Si hay datos disponibles, el usuario puede proceder con la búsqueda o reportar la lista actual.
- Proceso de Búsqueda:El usuario selecciona un criterio de búsqueda y pulsa el botón "Buscar".La aplicación ejecuta una consulta en la base de datos según criterios seleccionados. Si la consulta devuelve resultados, se muestran en el DGV. Si no hay resultados, se muestra un mensaje de "No se encontraron resultados" y el proceso finaliza.
- Generación de Reporte:Si hay datos en el DGV, el usuario puede pulsar el botón "Generar Reporte".La aplicación abre una nueva pantalla para mostrar el reporte generado.
- Exportación del Reporte:El usuario puede elegir exportar el reporte a diferentes formatos: PDF, Word, o Excel. Dependiendo del formato seleccionado, la aplicación genera el archivo correspondiente y lo muestra para descarga por el usuario.
- Finalización del Proceso: Después de generar y/o exportar el reporte, el usuario cierra la pantalla de reporte y el proceso finaliza.

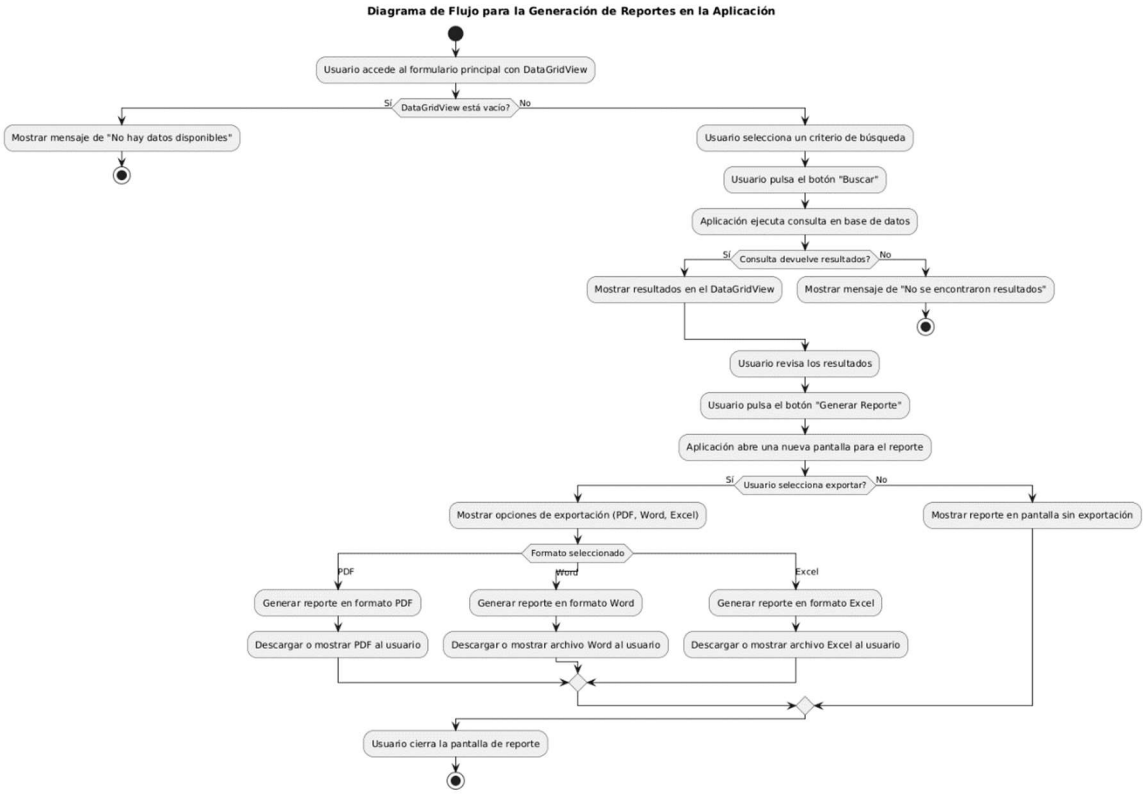


Figura 5.6 *Diagrama Reportes. Elaboración propia*

6. PRESUPUESTO Y ANÁLISIS FINANCIERO.

Para presupuestar el trabajo de desarrollo de una base de datos y aplicación de gestión de ofertas y proyectos en una empresa como AERTEC, es necesario considerar tanto la mano de obra como los materiales requeridos. A continuación, se desglosa un presupuesto generalizado en términos de recursos humanos y materiales, basado en una estimación de las horas efectivas requeridas y los profesionales necesarios para un proyecto de este tipo (véase Tabla 6.1).

6.1. RECURSOS HUMANOS

La mano de obra incluirá a todos los profesionales que participarán en el proyecto, como desarrolladores, analistas de sistemas, ingenieros de bases de datos, administradores de sistemas, y otros especialistas. Aquí hay una lista de roles típicos involucrados y el tiempo estimado de trabajo para cada uno:

- Jefe de Proyecto (40-60 horas): Gestión del proyecto, coordinación del equipo, planificación, seguimiento y control del progreso.
  - ✓ Definir el alcance del proyecto y crear el plan de proyecto.
  - ✓ Asignar tareas y asegurar que el equipo siga el cronograma.
  - ✓ Mantener comunicación con los superiores y con los miembros del equipo
  - ✓ Gestionar riesgos y cambios en los requisitos.
- Analista Informático (60-80 horas): Recolección y análisis de requisitos del cliente, documentación de especificaciones funcionales y no funcionales.
  - ✓ Entrevistar a los usuarios finales para entender las necesidades del proyecto.
  - ✓ Documentar los requisitos del sistema y crear especificaciones funcionales.
  - ✓ Validar que los desarrollos cumplen con los requisitos.
- Diseñador de Interfaz de Usuario (40-60 horas): Diseño de la interfaz de usuario y experiencia de usuario, creación de prototipos y maquetas.
  - ✓ Diseñar la interfaz de usuario asegurando una experiencia de usuario intuitiva.
  - ✓ Colaborar con desarrolladores para asegurar que el diseño se implemente correctamente.
- Arquitecto de Software (40-50 horas): Diseño de la arquitectura del sistema, selección de tecnologías y frameworks
  - ✓ Definir la arquitectura del sistema y las tecnologías a utilizar.
  - ✓ Crear diagramas de arquitectura y establecer estándares de codificación.
  - ✓ Revisar el código y asegurar la calidad del software.
- Desarrollador (80-120 horas): Desarrollo de la lógica de negocio e interfaz de usuario, integración con la base de datos.
  - ✓ Desarrollar la lógica de negocio del lado del servidor.
  - ✓ Crear e integrar servicios para la comunicación.
  - ✓ Implementar autenticación, autorización y validación de datos.
  - ✓ Implementar la interfaz de usuario basada en los diseños.
  - ✓ Crear componentes reutilizables y asegurar la interacción fluida entre MariaDB y Visual Estudio.
  - ✓ Asegurar la compatibilidad entre navegadores y optimización del rendimiento.
- Administrador de base de datos (60-80 horas): Diseño, implementación y gestión de la base de datos.
  - ✓ Crear y optimizar el esquema de la base de datos.

- ✓ Configurar la base de datos y asegurar la integridad de los datos.
- ✓ Implementar procedimientos almacenados, triggers y vistas

Este desglose es una estimación general para un proyecto típico con una aplicación CRUD, pero debe adaptarse según las necesidades y especificaciones de cada proyecto específico. La colaboración entre los diferentes perfiles es clave para asegurar que el proyecto se complete a tiempo, dentro del presupuesto y cumpliendo con los requisitos de calidad.

RRHH	Horas minimo	Horas Maximo	Precio horas	Salario Medio	Coste Proyecto Minimo	Coste Proyecto Maximo
Jefe Proyecto	40	60	11,872	32500	474,89	712,33
Analista Informatico	60	80	10,648	29150	638,90	851,87
Diseñador Gráfico	40	60	9,3699	25650	374,79	562,19
Arquitecto Software	40	50	10,046	27500	401,83	502,28
Desarrollador/Programador	80	120	10,082	27600	806,58	1209,86
Administrador en BD	60	80	9,8995	27100	593,97	791,96
Total RRHH					3290,96	4630,50
Licencias					0,00 €	0,00 €
equipo Informático					400,00 €	400,00 €
Total Proyecto					**3.690,96 €**	**5.030,50 €**

Tabla 6.1 *Presupuesto Base de Datos Elaboración Propia*

Para las tarifas he tenido en consideración el último proyecto del año 2023 para Junta Andalucía [9] donde se preciaban perfiles similares. Además de contrastar con:

- Informe de Remuneración 2023 de Hays
- INE (Instituto Nacional de Estadística
- Randstad Technologies Salary Guide

6.2. MATERIALES Y HERRAMIENTAS

El desarrollo del sistema también requiere una serie de herramientas y materiales, que pueden incluir:

- Licencias de Software: Para herramientas de desarrollo como Visual Studio, sistemas de gestión de bases de datos como MariaDB, y otras herramientas auxiliares de diseño y análisis, se ha utilizado las versiones gratuitas. Pero una vez se deba implanta sería conveniente actualizar las licencias profesionales de dichas aplicaciones, ya que tienen soporte.
- Hardware: Servidores para el desarrollo y pruebas, equipos de respaldo y almacenamiento para asegurar la disponibilidad y la integridad de los datos. Actualmente AERTEC dispone de servidores con capacidad para soportar esta aplicación.
- Infraestructura de Red: Configuración de red interna para facilitar la colaboración entre los equipos de desarrollo, pruebas y despliegue.
- Servicios en la Nube: Servicios adicionales como almacenamiento en la nube, bases de datos alojadas y servicios de backup para asegurar la continuidad del negocio y la protección de los datos. Tema pendiente a estudiar una vez se implante en más departamentos.

7. RESULTADOS Y DISCUSIÓN.

El sistema desarrollado podría incorporar funcionalidades avanzadas de control y seguimiento de proyectos. Dichas funcionalidades permitirían a los gestores de proyectos monitorizar el progreso de cada tarea, identificar posibles desviaciones y tomar decisiones informadas en tiempo real. Esto contribuye a mejorar la eficiencia operativa y a reducir los tiempos de entrega.

En términos de documentación y presentación, se ha elaborado la documentación técnica correspondiente, que incluye una descripción detallada de la base de datos, su estructura, los procedimientos de acceso y manipulación de los datos, así como cualquier otra información relevante (Anexo 10). Esta documentación es fundamental para garantizar una comprensión integral del sistema desarrollado y su correcta implementación y uso.

7.1. RENDIMIENTO AL CAMBIAR DE TABLAS EXCEL A BASE DE DATOS

El cambio de un sistema basado en hojas de cálculo de Excel a una base de datos puede resultar en mejoras significativas en eficiencia y reducción de costos. Los beneficios esperados se describen a continuación:

- Reducción de Tiempo en Gestión de Datos: La utilización de una base de datos permite una gestión más rápida y eficiente de la información, en comparación con la gestión manual en Excel.
- Disminución de Errores de Datos: El uso de una base de datos mejora la calidad de los datos gracias a la integridad referencial y las validaciones automáticas que previenen la entrada de datos incorrectos o duplicados. La centralización de la información en una base de datos mejora la consistencia y precisión de los datos, lo cual es crítico para la toma de decisiones estratégicas.
- Aumento de la Productividad del Equipo: La automatización de procesos y el acceso rápido a la información incrementan la productividad del equipo.
- Ahorro en Costos de Mantenimiento: La eliminación de la dependencia de múltiples hojas de cálculo y versiones reduce significativamente los costos asociados al mantenimiento del sistema.

Este cambio representa una inversión con beneficios significativos a largo plazo, proporcionando a AERTEC un sistema más robusto y eficiente, así como un marco que permite futuras expansiones e integraciones con otros sistemas empresariales.

7.2.USO DEL SOFTWARE "JAMA" Y RIESGOS PARA LA IMPLEMENTACIÓN DEL SISTEMA

Durante la estancia en AERTEC, se realizó una presentación del software "JAMA", del cual se han adquirido dos licencias de prueba con la intención de implementarlo en caso de que se demuestre su efectividad. Este software se utiliza para especificar y gestionar requisitos para proyectos de AIRBUS y otros clientes europeos, quienes exigen su uso como requisito para colaborar con ellos. El proyecto DEMO describe las necesidades de los usuarios, los requisitos de seguridad estándar, los riesgos, los requisitos de software, las pruebas de verificación y la arquitectura de la solución para simplificar la funcionalidad de la aplicación ReqView (Vista de Requisitos).

ReqView facilita a los usuarios documentar y gestionar requisitos, establecer trazabilidad entre estos, colaborar eficazmente y analizar el impacto de los cambios. A pesar de que las licencias y el mantenimiento de este software son costosos en comparación con el sistema diseñado, es una exigencia de los clientes que se use su plataforma para interactuar, lo cual también incluye la automatización del proceso de certificación en la entrega al cliente. Esto presenta un riesgo para la implementación en red de la base de datos diseñada, ya que podría no ser utilizada si el software "JAMA" se convierte en la solución predeterminada.

7.3. COMPARATIVA SECTOR AERONÁUTICO Y TELECOMUNICACIONES

Durante la estancia en AERTEC, se adquirió una mejor comprensión del sector aeronáutico, permitiendo realizar una comparación con el sector de las telecomunicaciones, donde se ha desarrollado una carrera profesional de más de 34 años. Ambos sectores presentan características distintivas en términos de márgenes de beneficio, relaciones con clientes y proveedores, y barreras de entrada.

- Márgenes de Beneficio: En el sector aeronáutico, los márgenes de beneficio son relativamente altos debido al costo significativo de los productos y servicios ofrecidos, aunque son volátiles como se observó durante la pandemia. La inversión en investigación y desarrollo (I+D) es considerable, a menudo subvencionada por fondos europeos o de la industria armamentística. Por el contrario, en telecomunicaciones, los márgenes son más ajustados y, aunque existe una férrea normativa, hay una alta competencia entre los diferentes operadores.
- Relaciones con Clientes: En el sector aeronáutico, las relaciones con los clientes tienden a ser a largo plazo, debido a la complejidad de los productos y servicios ofrecidos. En telecomunicaciones, las relaciones pueden variar desde contratos a largo plazo, especialmente en servicios corporativos, hasta relaciones más transaccionales para servicios de telecomunicaciones móviles.
- Relaciones con Proveedores: Los proveedores en el sector aeronáutico deben cumplir estrictos estándares de calidad y certificación, existiendo una alta dependencia de proveedores específicos y cualificados. En telecomunicaciones, hay una mayor diversidad de proveedores de hardware, software y servicios de red, aunque la dependencia de ciertos fabricantes de infraestructura de red puede ser significativa.
- Barreras de Entrada: Las barreras de entrada en el sector aeronáutico son extremadamente altas debido a la necesidad de grandes inversiones iniciales, altos costes de I+D, certificaciones estrictas y un mercado muy regulado. En telecomunicaciones, aunque también existen barreras significativas, estas son menores en comparación con el sector aeronáutico. La necesidad de inversión en infraestructura y licencias de espectro, junto con la regulación gubernamental, crean barreras para nuevos entrantes.

Ambos sectores requieren una inversión continua en innovación para mantener la competitividad. En telecomunicaciones, esta inversión se enfoca en áreas como 5G, fibra óptica, Internet de las Cosas (IoT), inteligencia artificial y servicios digitales, mientras que en aeronáutica se destina a mejorar la seguridad y eficiencia de los productos y servicios ofrecidos. Aunque el sector aeronáutico recibe más ayudas públicas, las telecomunicaciones impulsan la innovación en otros sectores como la salud, la educación y el comercio.

7.4. RECOMENDACIONES PARA LA IMPLEMENTACIÓN

Para mejorar la eficiencia y asegurar la sostenibilidad del sistema propuesto, se sugieren las siguientes recomendaciones:

- Mejoras en Seguridad: Se recomienda la implementación de medidas avanzadas de seguridad para proteger la integridad de los datos y prevenir accesos no autorizados. Esto incluye la encriptación de datos en reposo y en tránsito, así como la implementación de políticas de acceso basadas en roles.
- Optimización de Consultas SQL: Para mejorar el rendimiento de la base de datos, se sugiere la optimización de consultas SQL mediante el uso eficiente de índices, claves primarias y foráneas. Además, se recomienda realizar revisiones periódicas de las consultas y ajustar los índices según las necesidades cambiantes del sistema.
- Capacitación del Personal: Se recomienda proporcionar capacitación adecuada al personal involucrado en la gestión y uso del sistema, asegurando que comprendan completamente las capacidades y limitaciones del nuevo sistema de base de datos en comparación con el sistema anterior basado en Excel.
- Integración con Otros Sistemas: Considerar la posibilidad de integrar el sistema con otras plataformas utilizadas por los clientes, como el software "JAMA", para asegurar una interoperabilidad eficiente y cumplir con los requisitos específicos del cliente.
- Evaluación y Análisis Continuo: Se recomienda llevar a cabo una evaluación continua del sistema para identificar áreas de mejora y ajustar las estrategias según sea necesario para optimizar el rendimiento y maximizar los beneficios.

El cambio de un sistema basado en hojas de cálculo de Excel a una base de datos estructurada representa una mejora significativa en términos de eficiencia, reducción de errores, y ahorro de costos. Sin embargo, la necesidad de alinearse con las herramientas y plataformas requeridas por los clientes, como el software "JAMA", presenta desafíos que deben gestionarse cuidadosamente para asegurar el éxito del proyecto. A pesar de estos desafíos, la implementación de un sistema robusto y eficiente proporciona una base sólida para futuras expansiones e integraciones, asegurando la sostenibilidad y competitividad a largo plazo.

8. CONCLUSIONES.

En conclusión, el proyecto permitió la implementación de nuevas tecnologías y herramientas digitales en el departamento de sistemas no embarcados de AERTEC, alcanzando los objetivos principales de optimizar los procesos productivos y mejorar las relaciones con los clientes. A través de un enfoque estratégico de transformación digital, se consiguió no solo una gestión más estructurada y eficaz de la información, sino también un avance significativo en la modernización operativa de la empresa.

Durante la fase inicial de diagnóstico, se evaluó la situación de partida de la empresa y sus necesidades específicas. Este análisis fue crucial para identificar las áreas clave que requerían mejoras para avanzar hacia una digitalización más completa. Aunque AERTEC, como muchas empresas del sector, ya contaba con ciertos elementos digitales, el reto principal consistía en caracterizar sus productos y servicios y optimizar el uso de medios y canales digitales disponibles. Este proyecto permitió a la empresa avanzar en este sentido, estableciendo una base para futuras iniciativas de digitalización en otros departamentos.

La implementación de la nueva base de datos y las herramientas asociadas resultó en varios beneficios claros:

- Automatización de procesos: Se logró un mejor control de los datos, lo que incrementó la eficiencia operativa. La digitalización de la gestión de inventario y la trazabilidad de los productos redujo significativamente los errores humanos.
- Racionalización de los procedimientos: Se optimizaron los procedimientos operativos estándar, introduciendo mejoras continuas en la calidad del servicio. La capacidad de recopilar y analizar datos en tiempo real facilitó una toma de decisiones más informada y un seguimiento más preciso del desempeño operativo.
- Rediseño de procesos de negocio: El proyecto facilitó un análisis, simplificación y rediseño de los procesos existentes. Esto implicó una reorganización de los flujos de trabajo, la combinación de pasos redundantes y la eliminación de actividades repetitivas, lo que resultó en un modelo de operación más ágil y eficiente.
- Cambios de paradigma: Además de mejorar los procesos existentes, se motivó a la empresa a considerar nuevos modelos que se adapten mejor a un entorno digital. Este cambio puede fortalecer la relación con los clientes al ofrecer un valor añadido basado en un mayor conocimiento y una mejor respuesta a sus necesidades.
- El éxito del proyecto se atribuye a una combinación de un diagnóstico preciso de la situación actual, una estrategia clara de digitalización y un liderazgo efectivo, que impulsó el cambio cultural necesario para adoptar nuevas tecnologías y fomentar una mentalidad de mejora continua.

En definitiva, el proyecto cumplió con gran parte de los objetivos planteados desde su inicio. El camino hacia la transformación digital de AERTEC no termina aquí; este proyecto ha sentado las bases para futuras iniciativas que podrían extenderse a otros departamentos. AERTEC está ahora mejor posicionada para enfrentar los desafíos futuros en la industria aeroespacial, aprovechando las oportunidades que brinda la digitalización para mejorar sus procesos y modelos de negocio.

BIBLIOGRAFIA

[1] Codd, E. f. (1969). A Relational Model of Data for Large Shared Data Banks. 377-378.

[2] Manuel Enciso & Enrique Soler (Universidad de Málaga) Sublenguaje de Manipulación de datos de SQL.

[3] SAE International. (2010). ARP4754B: Guidelines for Development of Civil Aircraft and Systems. SAE International. https://www.sae.org/standards/content/arp4754b/

SAE International. (2018). ARP4761A: Guidelines and Methods for Conducting the Safety.

[4] Assessment Process on Civil Airborne Systems and Equipment. SAE International. https://www.sae.org/standards/content/arp4761a/

[5] RTCA. (2010). DO-160G: Environmental Conditions and Test Procedures for Airborne Equipment. RTCA, Inc. https://www.rtca.org/content/publications

[6] RTCA. (2011). DO-178C: Software Considerations in Airborne Systems and Equipment Certification. RTCA, Inc. https://www.rtca.org/content/publications

[7] RTCA. (2000). DO-254: Design Assurance Guidance for Airborne Electronic Hardware. RTCA, Inc. https://www.rtca.org/content/publications.

[8] MariaDB Foundation. (2024). *MariaDB Server*. From the MariaDB Knowledge Base, 2024-3. Ian Gilfillan. https://mariadb.com/kb/en/mariadb-documentation/

[9] EXPT22-0050: Servicio Soporte a Operaciones y Atención en Campo Especializado https://www.juntadeandalucia.es/haciendayadministracionpublica/apl/pdc_sirec/perfiles-licitaciones/detalle-licitacion.jsf?idExpediente=432816

https://www.youtube.com/@UPV

https://www.youtube.com/@mouredev

https://www.youtube.com/@HolaMundoDev

https://www.youtube.com/@puntojson

https://www.youtube.com/@devn8/playlists

https://espaciocompartir.inap.es/v3/course/

https://www.w3schools.com

https://www.manualweb.net/html5/

https://www.youtube.com/@VictorRamosDatSoft

https://chatgpt.com/auth/login

https://plantuml.com/es/

ANEXOS

ANEXO 1 NORMATIVAS ARP4754B, ARP4761A, DO-160, DO-178C, Y DO-254

Normativa	Título	Enfoque	Objetivos Principales
[3] ARP4754B	Guidelines for Development of Civil Aircraft and Systems	Desarrollo de sistemas aeronáuticos	- Gestión del ciclo de vida del sistema. - Ingeniería de sistemas y diseño. - Integración y evaluación de sistemas. - Aseguramiento de la seguridad de los sistemas.
[4] ARP4761A	Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment	Evaluación de la seguridad de sistemas aéreos	- Análisis de riesgos y peligros. - Métodos de evaluación de seguridad como FTA, FMEA, CCA. - Cumplimiento de requisitos de seguridad. - Coordinación con ARP4754B y otras normativas de seguridad.
[5] DO-160	Environmental Conditions and Test Procedures for Airborne Equipment	Pruebas ambientales y de equipos aéreos	- Definición de condiciones ambientales para pruebas. - Establecimiento de procedimientos de prueba para asegurar la resistencia del equipo a condiciones como temperatura, humedad, vibraciones, y radiación. - Cumplimiento de requisitos de certificación para equipos aéreos.
[6] DO-178C	Software Considerations in Airborne Systems and Equipment Certification	Certificación de software en sistemas aéreos	- Proceso de desarrollo de software seguro y fiable. - Establecimiento de niveles de garantía de diseño (DAL) para software. - Pruebas, verificación y validación del software. - Cumplimiento de requisitos de certificación de software aeronáutico.
[7] DO-254	Design Assurance Guidance for Airborne Electronic Hardware	Certificación de hardware electrónico en sistemas aéreos	- Proceso de desarrollo de hardware fiable para sistemas críticos. - Establecimiento de niveles de garantía de diseño (DAL) para hardware. - Requisitos para el diseño, verificación y validación del hardware. - Cumplimiento de certificación de hardware en sistemas aeronáuticos.

Figura A.1 *Resumen de los documentos técnicos que desarrollan estándares en la industria aeroespacial. Elaboración propia*

ANEXO 2 CÓDIGO ARCHIVO DDL GENERADO DATA MODELER

```
CREATE TABLE clientes (
    nombre_cliente    VARCHAR2(20) NOT NULL,
    departamento      VARCHAR2(20),
    direccion_cliente VARCHAR2(100)
);
ALTER TABLE clientes ADD CONSTRAINT clientes_pk PRIMARY KEY ( nombre_cliente );
CREATE TABLE ofertas (
    n_oferta              VARCHAR2(50) NOT NULL,
    titulo_oferta         VARCHAR2(100),
    fecha_oferta          DATE,
    proyecto_codigo_proyecto VARCHAR2(50) NOT NULL
);
```

```
ALTER TABLE ofertas ADD CONSTRAINT ofertas_pk PRIMARY KEY ( n_oferta,
                                                            proyecto_codigo_proyecto );
CREATE TABLE pedidos (
    n_pedido                NUMBER(20) NOT NULL,
    fecha_pedido            DATE,
    proyecto_codigo_proyecto VARCHAR2(50)
);
ALTER TABLE pedidos ADD CONSTRAINT pedidos_pk PRIMARY KEY ( n_pedido );
CREATE TABLE pn_n_serie (
    pn_cliente_serie VARCHAR2(100) NOT NULL,
    pn_aertec        VARCHAR2(100),
    descripci n      VARCHAR2(200),
    aclaratorio      VARCHAR2(200),
    tipo             VARCHAR2(11),
    programa         VARCHAR2(11),
    pn_cliente       VARCHAR2(100),
    n_serie          NUMBER(3),
    version          VARCHAR2(1)
);
ALTER TABLE pn_n_serie ADD CONSTRAINT pn_n_serie_pk PRIMARY KEY ( pn_cliente_serie );
CREATE TABLE proyecto (
    codigo_proyecto        VARCHAR2(50) NOT NULL,
    estado                 VARCHAR2(50),
    titulo_proyecto        VARCHAR2(100) NOT NULL,
    jefe_proyecto          VARCHAR2(100) NOT NULL,
    repositorio            VARCHAR2(100) NOT NULL,
    especificaciones       VARCHAR2(100),
    specific_qap           VARCHAR2(3) NOT NULL,
    wp_hardware            VARCHAR2(3),
    wp_manufacturing       VARCHAR2(3),
    wp_software            VARCHAR2(3),
    sw_repository          VARCHAR2(50),
    kpis                   CHAR(1),
    motivo                 VARCHAR2(100),
    clientes_nombre_cliente VARCHAR2(20)
);
ALTER TABLE proyecto ADD CONSTRAINT proyecto_pk PRIMARY KEY ( codigo_proyecto );
CREATE TABLE relation_14 (
    pn_n_serie_pn_cliente_serie VARCHAR2(100) NOT NULL,
    proyecto_codigo_proyecto    VARCHAR2(50) NOT NULL
);
ALTER TABLE relation_14 ADD CONSTRAINT relation_14_pk PRIMARY KEY ( pn_n_serie_pn_cliente_serie,
                                                                   proyecto_codigo_proyecto );
ALTER TABLE ofertas
    ADD CONSTRAINT ofertas_proyecto_fk FOREIGN KEY ( proyecto_codigo_proyecto )
        REFERENCES proyecto ( codigo_proyecto );
ALTER TABLE pedidos
    ADD CONSTRAINT pedidos_proyecto_fk FOREIGN KEY ( proyecto_codigo_proyecto )
        REFERENCES proyecto ( codigo_proyecto );
ALTER TABLE proyecto
    ADD CONSTRAINT proyecto_clientes_fk FOREIGN KEY ( clientes_nombre_cliente )
        REFERENCES clientes ( nombre_cliente );
ALTER TABLE relation_14
```

```
  ADD CONSTRAINT relation_14_pn_n_serie_fk FOREIGN KEY ( pn_n_serie_pn_cliente_serie )
    REFERENCES pn_n_serie ( pn_cliente_serie );
ALTER TABLE relation_14
  ADD CONSTRAINT relation_14_proyecto_fk FOREIGN KEY ( proyecto_codigo_proyecto )
    REFERENCES proyecto ( codigo_proyecto );
```

ANEXO 3 CÓDIGO PARA GENERAR TABLAS EN MARIA DB

```
CREATE TABLE IF NOT EXISTS `clientes` (
 `nombre_cliente` varchar(100) NOT NULL,
 `departamento` varchar(100) DEFAULT NULL,
 `direccion_cliente` varchar(100) DEFAULT NULL,
 PRIMARY KEY (`nombre_cliente`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
tabla pn_aertec.ofertas
CREATE TABLE IF NOT EXISTS `ofertas` (
 `n_oferta` varchar(50) NOT NULL,
 `titulo_oferta` varchar(100) NOT NULL,
 `fecha_oferta` date NOT NULL,
 `proyecto_codigo_proyecto` varchar(50) NOT NULL,
 PRIMARY KEY (`n_oferta`,`proyecto_codigo_proyecto`),
 KEY `ofertas_proyecto_fk` (`proyecto_codigo_proyecto`),
 CONSTRAINT `ofertas_proyecto_fk` FOREIGN KEY (`proyecto_codigo_proyecto`) REFERENCES `proyecto` (`codigo_proyecto`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
tabla pn_aertec.pedidos
CREATE TABLE IF NOT EXISTS `pedidos` (
 `n_pedido` varchar(50) NOT NULL,
 `fecha_pedido` date DEFAULT NULL,
 `proyecto_codigo_proyecto` varchar(50) DEFAULT NULL,
 PRIMARY KEY (`n_pedido`),
 KEY `pedidos_proyecto_fk` (`proyecto_codigo_proyecto`),
 CONSTRAINT `pedidos_proyecto_fk` FOREIGN KEY (`proyecto_codigo_proyecto`) REFERENCES `proyecto` (`codigo_proyecto`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
tabla pn_aertec.pn_n_serie
CREATE TABLE IF NOT EXISTS `pn_n_serie` (
 `pn_cliente_serie` varchar(100) NOT NULL,
 `pn_aertec` varchar(100) NOT NULL,
 `descripcion` varchar(200) NOT NULL,
 `aclaratorio` varchar(200) DEFAULT NULL,
 `tipo` varchar(11) DEFAULT NULL,
 `programa` varchar(11) DEFAULT NULL,
 `n_serie` varchar(50) DEFAULT NULL,
 `version` varchar(3) DEFAULT NULL,
 PRIMARY KEY (`pn_cliente_serie`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
tabla pn_aertec.proyecto
CREATE TABLE IF NOT EXISTS `proyecto` (
 `codigo_proyecto` varchar(50) NOT NULL,
 `estado` varchar(50) NOT NULL,
 `titulo_proyecto` varchar(100) NOT NULL,
 `jefe_proyecto` varchar(100) NOT NULL,
 `repositorio` varchar(50) NOT NULL,
```

```
  `especificaciones` varchar(50) NOT NULL,
  `specific_qap` varchar(3) NOT NULL,
  `wp_hardware` varchar(3) NOT NULL,
  `wp_manufacturing` varchar(3) NOT NULL,
  `wp_software` varchar(3) NOT NULL,
  `sw_repository` varchar(50) NOT NULL,
  `kpis` varchar(3) NOT NULL,
  `motivo` varchar(50) NOT NULL,
  `clientes_nombre_cliente` varchar(100) NOT NULL,
  PRIMARY KEY (`codigo_proyecto`),
  KEY `FK1CLIENTES` (`clientes_nombre_cliente`),
  CONSTRAINT `FK1CLIENTES` FOREIGN KEY (`clientes_nombre_cliente`) REFERENCES `clientes` (`nombre_cliente`) ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
tabla pn_aertec.relation_14
CREATE TABLE IF NOT EXISTS `relation_14` (
  `pn_n_serie_pn_cliente_serie` varchar(100) NOT NULL,
  `proyecto_codigo_proyecto` varchar(50) NOT NULL,
  PRIMARY KEY (`pn_n_serie_pn_cliente_serie`,`proyecto_codigo_proyecto`),
  KEY `relation_14_proyecto_fk` (`proyecto_codigo_proyecto`),
  CONSTRAINT `relation_14_pn_n_serie_fk` FOREIGN KEY (`pn_n_serie_pn_cliente_serie`) REFERENCES `pn_n_serie` (`pn_cliente_serie`),
  CONSTRAINT `relation_14_proyecto_fk` FOREIGN KEY (`proyecto_codigo_proyecto`) REFERENCES `proyecto` (`codigo_proyecto`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_spanish_ci;
DELIMITER ;
```

ANEXO 4 PROCEDIMIENTOS ALMACENADOS EN MARIA DB

```
procedimiento pn_aertec.ActualizarCliente
DELIMITER //
CREATE PROCEDURE `ActualizarCliente`(
    IN p_nombre_cliente VARCHAR(20),
    IN p_departamento VARCHAR(20),
    IN p_direccion_cliente VARCHAR(100)
)
BEGIN
    UPDATE clientes
    SET departamento = p_departamento,
        direccion_cliente = p_direccion_cliente
    WHERE nombre_cliente = p_nombre_cliente;
END//
DELIMITER ;
 *procedimiento pn_aertec.ActualizarOferta
DELIMITER //
CREATE PROCEDURE `ActualizarOferta`(
    IN p_n_oferta VARCHAR(50),
    IN p_titulo_oferta VARCHAR(100),
    IN p_fecha_oferta DATE,
    IN p_proyecto_codigo_proyecto VARCHAR(50)
)
BEGIN
    UPDATE ofertas
    SET titulo_oferta = p_titulo_oferta,
        fecha_oferta = p_fecha_oferta
    WHERE n_oferta = p_n_oferta
      AND proyecto_codigo_proyecto = p_proyecto_codigo_proyecto;
```

```
END//
DELIMITER ;
 *procedimiento pn_aertec.ActualizarPnNSerie
DELIMITER //
CREATE PROCEDURE `ActualizarPnNSerie`(
    IN p_pn_cliente_serie VARCHAR(100),
    IN p_pn_aertec VARCHAR(100),
    IN p_descripcion VARCHAR(200),
    IN p_aclaratorio VARCHAR(200),
    IN p_tipo VARCHAR(11),
    IN p_programa VARCHAR(11),
    IN p_n_serie VARCHAR(50),
    IN p_version VARCHAR(3)
)
BEGIN
    UPDATE pn_n_serie
    SET pn_aertec = p_pn_aertec,
        descripción= p_descripcion,
        aclaratorio = p_aclaratorio,
        tipo = p_tipo,
        programa = p_programa,
        n_serie = p_n_serie,
        versión = p_version
    WHERE pn_cliente_serie = p_pn_cliente_serie;
END//
DELIMITER ;
 *procedimiento pn_aertec.ActualizarProyecto
DELIMITER //
CREATE PROCEDURE `ActualizarProyecto`(
    IN p_codigo_proyecto VARCHAR(50),
    IN p_estado VARCHAR(50),
    IN p_titulo_proyecto VARCHAR(100),
    IN p_jefe_proyecto VARCHAR(100),
    IN p_repositorio VARCHAR(50),
    IN p_especificaciones VARCHAR(50),
    IN p_specific_qap varCHAR(3),
    IN p_wp_hardware varCHAR(3),
    IN p_wp_manufacturing varCHAR(3),
    IN p_wp_software varCHAR(3),
    IN p_sw_repository VARCHAR(50),
    IN p_kpis varCHAR(3),
    IN p_motivo VARCHAR(50),
    IN p_clientes_nombre_cliente VARCHAR(100)
)
BEGIN
    UPDATE proyecto
    SET estado = p_estado,
        titulo_proyecto = p_titulo_proyecto,
        jefe_proyecto = p_jefe_proyecto,
        repositorio = p_repositorio,
        especificaciones = p_especificaciones,
        specific_qap = p_specific_qap,
        wp_hardware = p_wp_hardware,
```

```
        wp_manufacturing = p_wp_manufacturing,
        wp_software = p_wp_software,
        sw_repository = p_sw_repository,
        kpis = p_kpis,
        motivo = p_motivo,
        clientes_nombre_cliente = p_clientes_nombre_cliente
    WHERE codigo_proyecto = p_codigo_proyecto;
END//
DELIMITER ;
 *procedimiento pn_aertec.EliminarCliente
DELIMITER //
CREATE PROCEDURE `EliminarCliente`(
    IN p_nombre_cliente VARCHAR(20)
)
BEGIN
    DELETE FROM clientes WHERE nombre_cliente = p_nombre_cliente;
END//
DELIMITER ;
 *procedimiento pn_aertec.EliminarOferta
DELIMITER //
CREATE PROCEDURE `EliminarOferta`(
    IN p_n_oferta VARCHAR(50),
    IN p_proyecto_codigo_proyecto VARCHAR(50)
)
BEGIN
    DELETE FROM ofertas
    WHERE n_oferta = p_n_oferta
      AND proyecto_codigo_proyecto = p_proyecto_codigo_proyecto;
END//
DELIMITER ;
 *procedimiento pn_aertec.EliminarPedido
DELIMITER //
CREATE PROCEDURE `EliminarPedido`(
    IN p_n_pedido VARCHAR (50)
)
BEGIN
    DELETE FROM pedidos WHERE n_pedido = p_n_pedido;
END//
DELIMITER ;
 *procedimiento pn_aertec.EliminarPnNSerie
DELIMITER //
CREATE PROCEDURE `EliminarPnNSerie`(
    IN p_pn_cliente_serie VARCHAR(100)
)
BEGIN
    DELETE FROM pn_n_serie WHERE pn_cliente_serie = p_pn_cliente_serie;
END//
DELIMITER ;
 *procedimiento pn_aertec.EliminarProyecto
DELIMITER //
CREATE PROCEDURE `EliminarProyecto`(
    IN p_codigo_proyecto VARCHAR(50)
)
```

```
BEGIN
    DELETE FROM proyecto WHERE codigo_proyecto = p_codigo_proyecto;
END//
DELIMITER ;
 *procedimiento pn_aertec.InsertarCliente
DELIMITER //
CREATE PROCEDURE `InsertarCliente`(
    IN p_nombre_cliente VARCHAR(20),
    IN p_departamento VARCHAR(20),
    IN p_direccion_cliente VARCHAR(100)
)
BEGIN
    INSERT INTO clientes (nombre_cliente, departamento, direccion_cliente)
    VALUES (p_nombre_cliente, p_departamento, p_direccion_cliente);
END//
DELIMITER ;
 *procedimiento pn_aertec.InsertarOferta
DELIMITER //
CREATE PROCEDURE `InsertarOferta`(
    IN p_n_oferta VARCHAR(50),
    IN p_titulo_oferta VARCHAR(100),
    IN p_fecha_oferta DATE,
    IN p_proyecto_codigo_proyecto VARCHAR(50)
)
BEGIN
    INSERT INTO ofertas (n_oferta, titulo_oferta, fecha_oferta, proyecto_codigo_proyecto)
    VALUES (p_n_oferta, p_titulo_oferta, p_fecha_oferta, p_proyecto_codigo_proyecto);
END//
DELIMITER ;
 *procedimiento pn_aertec.InsertarPedido
DELIMITER //
CREATE PROCEDURE `InsertarPedido`(
    IN p_n_pedido VARCHAR(50),
    IN p_fecha_pedido DATE,
    IN p_proyecto_codigo_proyecto VARCHAR(50)
)
BEGIN
    INSERT INTO pedidos (n_pedido, fecha_pedido, proyecto_codigo_proyecto)
    VALUES (p_n_pedido, p_fecha_pedido, p_proyecto_codigo_proyecto);
END//
DELIMITER ;
 *procedimiento pn_aertec.InsertarPnNSerie
DELIMITER //
CREATE PROCEDURE `InsertarPnNSerie`(
    IN p_pn_cliente_serie VARCHAR(100),
    IN p_pn_aertec VARCHAR(100),
    IN p_descripcion VARCHAR(200),
    IN p_aclaratorio VARCHAR(200),
    IN p_tipo VARCHAR(11),
    IN p_programa VARCHAR(11),
    IN p_n_serie VARCHAR(4),
    IN p_version VARCHAR(1)
)
```

```
BEGIN
    INSERT INTO pn_n_serie (pn_cliente_serie, pn_aertec, descripción, aclaratorio, tipo, programa, n_serie, versión)
    VALUES (p_pn_cliente_serie, p_pn_aertec, p_descripcion, p_aclaratorio, p_tipo, p_programa, p_n_serie, p_version);
END//
DELIMITER ;
 *procedimiento pn_aertec.InsertarProyecto
DELIMITER //
CREATE PROCEDURE `InsertarProyecto`(
    IN p_codigo_proyecto VARCHAR(50),
    IN p_estado VARCHAR(50),
    IN p_titulo_proyecto VARCHAR(100),
    IN p_jefe_proyecto VARCHAR(100),
    IN p_repositorio VARCHAR(50),
    IN p_especificaciones VARCHAR(50),
    IN p_specific_qap VARCHAR(3),
    IN p_wp_hardware varCHAR(3),
    IN p_wp_manufacturing VARCHAR(3),
    IN p_wp_software VARCHAR(3),
    IN p_sw_repository VARCHAR(50),
    IN p_kpis VARCHAR(3),
    IN p_motivo VARCHAR(50),
    IN p_clientes_nombre_cliente VARCHAR(100)
)
BEGIN
    INSERT INTO proyecto (codigo_proyecto, estado, titulo_proyecto, jefe_proyecto, repositorio, especificaciones,
                specific_qap, wp_hardware, wp_manufacturing, wp_software, sw_repository, kpis, motivo, clientes_nombre_cliente)
    VALUES (p_codigo_proyecto, p_estado, p_titulo_proyecto, p_jefe_proyecto, p_repositorio, p_especificaciones,
        p_specific_qap, p_wp_hardware, p_wp_manufacturing, p_wp_software, p_sw_repository, p_kpis, p_motivo, p_clientes_nombre_cliente);
END//
DELIMITER ;
 *procedimiento pn_aertec.ListarClientes
DELIMITER //
CREATE PROCEDURE `ListarClientes`(
IN `Nombre` VARCHAR(50)
)
BEGIN
    SELECT * FROM clientes;
END//
DELIMITER ;
 *procedimiento pn_aertec.ListarOfertas
DELIMITER //
CREATE PROCEDURE `ListarOfertas`()
BEGIN
    SELECT * FROM ofertas;
END//
DELIMITER ;
 *procedimiento pn_aertec.ListarPedidos
DELIMITER //
CREATE PROCEDURE `ListarPedidos`()
BEGIN
    SELECT * FROM pedidos;
END//
DELIMITER ;
```

```
 *procedimiento pn_aertec.ListarPnNSerie
DELIMITER //
CREATE PROCEDURE `ListarPnNSerie`()
BEGIN
    SELECT * FROM pn_n_serie;
END//
DELIMITER ;
 *procedimiento pn_aertec.ListarProyectos
DELIMITER //
CREATE PROCEDURE `ListarProyectos`(
IN `cTexto` VARCHAR(50)
)
BEGIN
    SELECT * FROM proyecto;
END//
DELIMITER ;
 *procedimiento pn_aertec.ObtenerClientePorNombre
DELIMITER //
CREATE PROCEDURE `ObtenerClientePorNombre`(
    IN p_nombre_cliente VARCHAR(20)
)
BEGIN
    SELECT * FROM clientes WHERE nombre_cliente = p_nombre_cliente;
END//
DELIMITER ;
 *procedimiento pn_aertec.ObtenerOfertaPorID
DELIMITER //
CREATE PROCEDURE `ObtenerOfertaPorID`(
    IN p_n_oferta VARCHAR(50),
    IN p_proyecto_codigo_proyecto VARCHAR(50)
)
BEGIN
    SELECT * FROM ofertas
    WHERE n_oferta = p_n_oferta
      AND proyecto_codigo_proyecto = p_proyecto_codigo_proyecto;
END//
DELIMITER ;
 *procedimiento pn_aertec.ObtenerProyectoPorCodigo
DELIMITER //
CREATE PROCEDURE `ObtenerProyectoPorCodigo`(
    IN p_codigo_proyecto VARCHAR(50)
)
BEGIN
    SELECT * FROM proyecto WHERE codigo_proyecto = p_codigo_proyecto;
END//
DELIMITER ;
```

ANEXO 5 CÓDIGO FORMULARIO PROYECTOS VARIABLES Y MÉTODOS.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
namespace MTD_CRUD_PN.PRESENTACION
{
    public partial class frmProyectos : Form
    {
        public frmProyectos()
        {
            InitializeComponent();
        }
        #region "Mis Variables"
        int nEstadoguarda = 0;
        int nespecificaciones = 0;
        int ncodigo_proyecto = 0;
        int nEstado = 0;
        int ntitulo_proyecto = 0;
        int nrepositorio = 0;
        int njefe_proyecto = 0;
        int nspecific_qap = 0;
        int nwp_hardware = 0;
        int nwp_manufacturing = 0;
        int nwp_software = 0;
        int nsw_repository = 0;
        int nKPIs = 0;
        int nmotivo = 0;


        #endregion

        #region "Mis metodos"

        private void Formato()
        {
            dgvProyectos.Columns[0].Width = 50;
            dgvProyectos.Columns[0].HeaderText = "Código Proyecto";
            dgvProyectos.Columns[1].Width = 50;
            dgvProyectos.Columns[1].HeaderText = "Estado";
            dgvProyectos.Columns[2].Width = 100;
            dgvProyectos.Columns[2].HeaderText = "Titulo proyecto";
            dgvProyectos.Columns[3].Width = 100;
            dgvProyectos.Columns[3].HeaderText = "Jefe Proyecto";
            dgvProyectos.Columns[4].Width = 100;
            dgvProyectos.Columns[4].HeaderText = "Repositorio";
            dgvProyectos.Columns[5].Width = 100;
            dgvProyectos.Columns[5].HeaderText = "especificaciones";
            dgvProyectos.Columns[6].Width = 30;
            dgvProyectos.Columns[6].HeaderText = "specific_qap";
            dgvProyectos.Columns[7].Width = 30;
            dgvProyectos.Columns[7].HeaderText = "wp_hardware";
            dgvProyectos.Columns[8].Width = 30;
```

```
    dgvProyectos.Columns[8].HeaderText = "wp_manufacturing";
    dgvProyectos.Columns[9].Width = 30;
    dgvProyectos.Columns[9].HeaderText = "wp_software";
    dgvProyectos.Columns[10].Width = 50;
    dgvProyectos.Columns[10].HeaderText = "sw_repository";
    dgvProyectos.Columns[11].Width = 3;
    dgvProyectos.Columns[11].HeaderText = "KPIs";
    dgvProyectos.Columns[12].Width = 50;
    dgvProyectos.Columns[12].HeaderText = "Motivo";
    dgvProyectos.Columns[13].Width = 100;
    dgvProyectos.Columns[13].HeaderText = "Cliente";
}
private void Listado_Proyecto(string cTexto)
{
    D_Proyectos Datos = new D_Proyectos();
    dgvProyectos.DataSource = Datos.Listado_Proyecto(cTexto);
    this.Formato();
}
private void Limpia_texto()
{
    txtcodproyecto.Text = "";
    txtestado.Text = "";
    txtproyecto.Text = "";
    txtjefeproyecto.Text = "";
    txtespecificaciones.Text = "";
    txtspecific_qap.Text = "";
    txtsw_repository.Text = "";
    txtwp_hardware.Text = "";
    txtwp_manufacturing.Text = "";
    txtwp_software.Text = "";
    txtrepositorio.Text = "";
    txtKPIs.Text = "";
    txtmotivo.Text = "";
    cmbcliente.Text = "";
}

private void Estado_texto(bool lEstado)
{
    txtcodproyecto.Enabled = lEstado;
    txtestado.Enabled = lEstado;
    txtproyecto.Enabled = lEstado;
    txtjefeproyecto.Enabled = lEstado;
    txtespecificaciones.Enabled = lEstado;
    txtspecific_qap.Enabled = lEstado;
    txtsw_repository.Enabled = lEstado;
    txtwp_hardware.Enabled = lEstado;
    txtwp_manufacturing.Enabled = lEstado;
    txtwp_software.Enabled = lEstado;
    txtrepositorio.Enabled = lEstado;
    txtKPIs.Enabled = lEstado;
    txtmotivo.Enabled = lEstado;
    cmbcliente.Enabled = lEstado;
}
```

```
private void Estado_BotonesProcesos(bool lEstado)
{
    btncancelar.Visible = lEstado;
    btnguardar.Visible = lEstado;
}
private void Estado_BotonesPrincipales(bool lEstado)
{
    btn_nuevo.Enabled = lEstado;
    btn_actualizar.Enabled = lEstado;
    btn_eliminar.Enabled = lEstado;
    btn_reporte.Enabled = lEstado;
    btn_salir.Enabled = lEstado;
}

private void Seleccion_codigo_proyecto()
{
    if (string.IsNullOrEmpty(Convert.ToString(dgvProyectos.CurrentRow.Cells["codproyecto"].Value)))
    {
        MessageBox.Show("Seleccione un  registro",
                        "Aviso del Sistema",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Exclamation);
    }
    else
    {
        txtcodproyecto.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["CODIGO_PROYECTO"].Value);
        txtestado.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["ESTADO"].Value);
        txtproyecto.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["TITULO_PROYECTO"].Value);
        txtjefeproyecto.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["JEFE_PROYECTO"].Value);
        txtespecificaciones.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["specific_qap"].Value);
        txtwp_hardware.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["wp_hardware"].Value);
        txtwp_manufacturing.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["wp_manufacturing"].Value);
        txtwp_software.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["wp_software"].Value);
        txtrepositorio.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["sw_repository"].Value);
        txtKPIs.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["KPIs"].Value);
        txtmotivo.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["MOTIVO"].Value);
    }
}
#endregion
private void btn_nuevo_Click(object sender, EventArgs e)
{
    int nEstadoguarda = 1;  // Nuevo registro
    this.Limpia_texto();
    this.Estado_texto(true);
    this.Estado_BotonesProcesos(true);
    this.Estado_BotonesPrincipales(false);
    txtcodproyecto.Focus();
}
private void btn_salir_Click_1(object sender, EventArgs e)
{
    this.Close();
}
private void btn_cancelar_Click_1(object sender, EventArgs e)
```

```
{
    this.Limpia_texto();
    this.Estado_texto(false);
    this.Estado_BotonesProcesos(false);
    this.Estado_BotonesPrincipales(true);
}
private void Formproyectos_Load(object sender, EventArgs e)
{
    this.Listado_Proyecto("%");
}
private void btn_actualizar_Click(object sender, EventArgs e)
{
    this.Estado_texto(true);
    this.Estado_BotonesProcesos(true);
    this.Estado_BotonesPrincipales(false);
    txtcodproyecto.Focus();
}
private void btn_eliminar_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(Convert.ToString(dgvProyectos.CurrentRow.Cells["codproyecto"].Value)))
    {
        MessageBox.Show("Seleccione un registro",
                        "Aviso del Sistema",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Exclamation);
    }
    else
    {
        DialogResult Opcion;
        Opcion = MessageBox.Show("¿Realmente deseas eliminar el registro?",
                                 "Aviso del Sistema",
                                 MessageBoxButtons.OKCancel,
                                 MessageBoxIcon.Question);
        if (Opcion == DialogResult.OK)
        {
            string Rpta = "";
            int idProyecto = Convert.ToInt32(dgvProyectos.CurrentRow.Cells["codproyecto"].Value);
            D_Proyectos Datos = new D_Proyectos();
            Rpta = Datos.EliminarProyecto(idProyecto);
            if (Rpta.Equals("OK"))
            {
                this.Listado_Proyecto("%");
                MessageBox.Show("Registro eliminado correctamente",
                                "Aviso del Sistema",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show(Rpta,
                                "Aviso del Sistema",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Error);
```

```
        }
      }
    }
}
private void frmProyectos_Load(object sender, EventArgs e)
{
    this.Listado_Proyecto("%");
}
private void btn_nuevo_Click_1(object sender, EventArgs e)
{
    this.Limpia_texto();
    this.Estado_texto(true);
    this.Estado_BotonesProcesos(true);
    this.Estado_BotonesPrincipales(false);
    txtcodproyecto.Focus();
}
private void btn_salir_Click(object sender, EventArgs e)
{
    this.Close();
}

private void btncancelar_Click(object sender, EventArgs e)
{
    this.Limpia_texto();
    this.Estado_texto(false);
    this.Estado_BotonesProcesos(false);
    this.Estado_BotonesPrincipales(true);
}

private void btnbuscar_Click(object sender, EventArgs e)
{
    this.Listado_Proyecto(txtbuscar.Text);
}
private void btn_actualizar_Click_1(object sender, EventArgs e)
{
}
private void dgvProyectos_CellClick(object sender, DGVCellEventArgs e)
{
    if (string.IsNullOrEmpty(Convert.ToString(dgvProyectos.CurrentRow.Cells["codigo_proyecto"].Value)))
    {
        MessageBox.Show("Seleccione un  registro",
                        "Aviso del Sistema",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Exclamation);
    }
    else
    {
        txtcodproyecto.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["CODIGO_PROYECTO"].Value);
        txtestado.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["ESTADO"].Value);
        txtproyecto.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["TITULO_PROYECTO"].Value);
        txtjefeproyecto.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["JEFE_PROYECTO"].Value);
        txtrepositorio.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["Repositorio"].Value);
        txtespecificaciones.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["Especificaciones"].Value);
```

```
            txtspecific_qap.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["specific_qap"].Value);
            txtwp_hardware.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["wp_hardware"].Value);
            txtwp_manufacturing.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["wp_manufacturing"].Value);
            txtwp_software.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["wp_software"].Value);
            txtsw_repository.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["sw_repository"].Value);
            txtKPIs.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["KPIs"].Value);
            txtmotivo.Text = Convert.ToString(dgvProyectos.CurrentRow.Cells["MOTIVO"].Value);
        }
    }
    private void btnguardar_Click(object sender, EventArgs e)
    {
        // Crear una instancia de la capa de datos
        D_Proyectos dataAccess = new D_Proyectos();
        // Obtener los datos ingresados por el usuario
        string codigoProyecto = txtcodproyecto.Text;
        string estado = txtestado.Text;
        string tituloProyecto = txtproyecto.Text;
        string jefeProyecto = txtjefeproyecto.Text;
        string repositorio = txtrepositorio.Text;
        string especificaciones = txtespecificaciones.Text;
        string specificQap = txtspecific_qap.Text;
        string wpHardware = txtwp_hardware.Text;
        string wpManufacturing = txtwp_manufacturing.Text;
        string wpSoftware = txtwp_software.Text;
        string swRepository = txtsw_repository.Text;
        string kpis = txtKPIs.Text;
        string motivo = txtmotivo.Text;
        string clientesNombreCliente = "";
        // Llamar al método para insertar el proyecto
        bool resultado = dataAccess.InsertarProyecto(codigoProyecto, estado, tituloProyecto, jefeProyecto, repositorio,
                                        especificaciones, specificQap, wpHardware, wpManufacturing,
                                        wpSoftware, swRepository, kpis, motivo, clientesNombreCliente);
        // Verificar el resultado
        if (resultado)
        {
            MessageBox.Show("Proyecto insertado exitosamente.");
        }
        else
        {
            MessageBox.Show("Error al insertar el proyecto.");
        }
    }
  }

}
```

ANEXO 6 CÓDIGO D_PROYECTOS.

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;
namespace MTD_CRUD_PN.DATOS
{
    public class D_Proyectos
    {

        public DataTable Listado_Proyecto(string cTexto)
        {
            MySqlDataReader Resultado;
            DataTable Tabla = new DataTable();
            MySqlConnection sqlCon = new MySqlConnection();
            try
            {
                sqlCon = conexion.getInstancia().CrearConexion();
                MySqlCommand Comando = new MySqlCommand("ListarProyectos", sqlCon);
                Comando.CommandType = CommandType.StoredProcedure;
                Comando.Parameters.Add("cTexto", MySqlDbType.VarChar).Value = cTexto;
                sqlCon.Open();
                Resultado = Comando.ExecuteReader();
                Tabla.Load(Resultado);
                return Tabla;
            }
            catch (Exception ex)
            {
                throw ex;
            }
            finally
            {
                if (sqlCon.State == ConnectionState.Open) sqlCon.Close();
            }
        }
        public string EliminarProyecto(int codigo_proyecto)
        {
            string Rpta = "";
            MySqlConnection SqlCon = new MySqlConnection();
            try
            {
                SqlCon = conexion.getInstancia().CrearConexion();
                MySqlCommand Comando = new MySqlCommand("EliminarProyecto", SqlCon);
                Comando.CommandType = CommandType.StoredProcedure;
                Comando.Parameters.Add("p_codigo_proyecto", MySqlDbType.VarChar).Value = codigo_proyecto;
                SqlCon.Open();
                Rpta = Comando.ExecuteNonQuery() >= 1 ? "OK" : "No se puede eliminar el registro";
            }
            catch (Exception ex)
            {
                Rpta = ex.Message;
            }
            finally
            {
                if (SqlCon.State == ConnectionState.Open) SqlCon.Close();
            }
```

```
        return Rpta;
    }
    // Cadena de conexión a la base de datos MariaDB
    private string connectionString = "Server=localhost;Database=pn_aertec;Puerto=3306;User ID=user_pn;Password=Maravila16;";
    public bool InsertarProyecto(string codigoProyecto, string estado, string tituloProyecto, string jefeProyecto,
                                 string repositorio, string especificaciones, string specificQap, string wpHardware,
                                 string wpManufacturing, string wpSoftware, string swRepository, string kpis,
                                 string motivo, string clientesNombreCliente)
    {
        using (MySqlConnection connection = new MySqlConnection(connectionString))
        {
            try
            {
                // Abrir la conexión a la base de datos
                connection.Open();
                // Crear un comando para ejecutar el procedimiento almacenado
                MySqlCommand cmd = new MySqlCommand("InsertarProyecto", connection);
                cmd.CommandType = CommandType.StoredProcedure;
                // Agregar los parámetros del procedimiento almacenado
                cmd.Parameters.AddWithValue("@p_codigo_proyecto", codigoProyecto);
                cmd.Parameters.AddWithValue("@p_estado", estado);
                cmd.Parameters.AddWithValue("@p_titulo_proyecto", tituloProyecto);
                cmd.Parameters.AddWithValue("@p_jefe_proyecto", jefeProyecto);
                cmd.Parameters.AddWithValue("@p_repositorio", repositorio);
                cmd.Parameters.AddWithValue("@p_especificaciones", especificaciones);
                cmd.Parameters.AddWithValue("@p_specific_qap", specificQap);
                cmd.Parameters.AddWithValue("@p_wp_hardware", wpHardware);
                cmd.Parameters.AddWithValue("@p_wp_manufacturing", wpManufacturing);
                cmd.Parameters.AddWithValue("@p_wp_software", wpSoftware);
                cmd.Parameters.AddWithValue("@p_sw_repository", swRepository);
                cmd.Parameters.AddWithValue("@p_kpis", kpis);
                cmd.Parameters.AddWithValue("@p_motivo", motivo);
                cmd.Parameters.AddWithValue("@p_clientes_nombre_cliente", clientesNombreCliente);
                // Ejecutar el comando
                cmd.ExecuteNonQuery();
                return true; // Operación exitosa
            }
            catch (Exception ex)
            {
                // Manejo de errores
                MessageBox.Show("Error al insertar proyecto: " + ex.Message);
                return false; // Operación fallida
            }
        }
    }
}
```

ANEXO 7 CÓDIGO C# FORMULARIO DE ACCESO

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace MTD_CRUD_PN.PRESENTACION
{
  public partial class LoginForm : Form
  {
    public LoginForm()
    {
      InitializeComponent();
    }
    private void btnLogin_Click(object sender, EventArgs e)
    {
      string username = txtusuario.Text;
      string password = txtcontraseña.Text;
      if (string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
      {
        lblMensaje.Text = "Por favor, ingrese su nombre de usuario y contraseña.";
        return;
      }
      try
      {
        // Verifica las credenciales del usuario
        if (ValidarUsuario(username, password))
        {
          MessageBox.Show("Acceso concedido", "Éxito", MessageBoxButtons.OK, MessageBoxIcon.Information);
          this.Hide(); // Ocultar el formulario de login
          //MainForm mainForm = new MainForm(); // Asume que tienes un formulario principal llamado MainForm
          //mainForm.Show();
        }
        else
        {
          lblMensaje.Text = "Nombre de usuario o contraseña incorrectos.";
        }
      }
      catch (Exception ex)
      {
        MessageBox.Show("Error al conectar con la base de datos: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
    }
    private bool ValidarUsuario(string username, string password)
    {
      bool isValid = false;
      string connectionString = "Server=localhost;Database=tu_base_de_datos;User ID=tu_usuario;Password=tu_contraseña;";
      using (MySqlConnection connection = new MySqlConnection(connectionString))
      {
        string query = "SELECT COUNT(1) FROM usuarios WHERE username=@username AND password=@password";
        MySqlCommand cmd = new MySqlCommand(query, connection);
        cmd.Parameters.AddWithValue("@username", username);
        cmd.Parameters.AddWithValue("@password", password);
```

```
            connection.Open();
            int count = Convert.ToInt32(cmd.ExecuteScalar());
            if (count == 1)
            {
                isValid = true;
            }
            connection.Close();
        }
        return isValid;
    }
  }
}
```

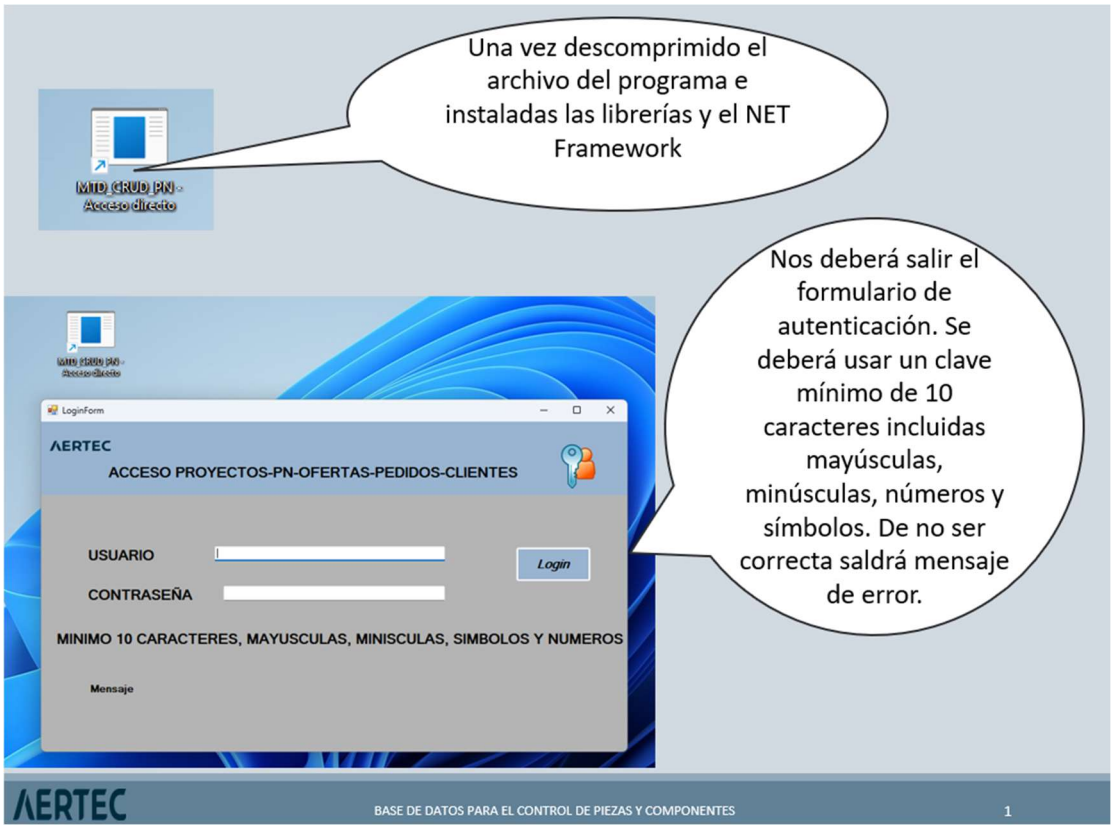
ANEXO 8 CÓDIGO CONEXIÓN MARIADB

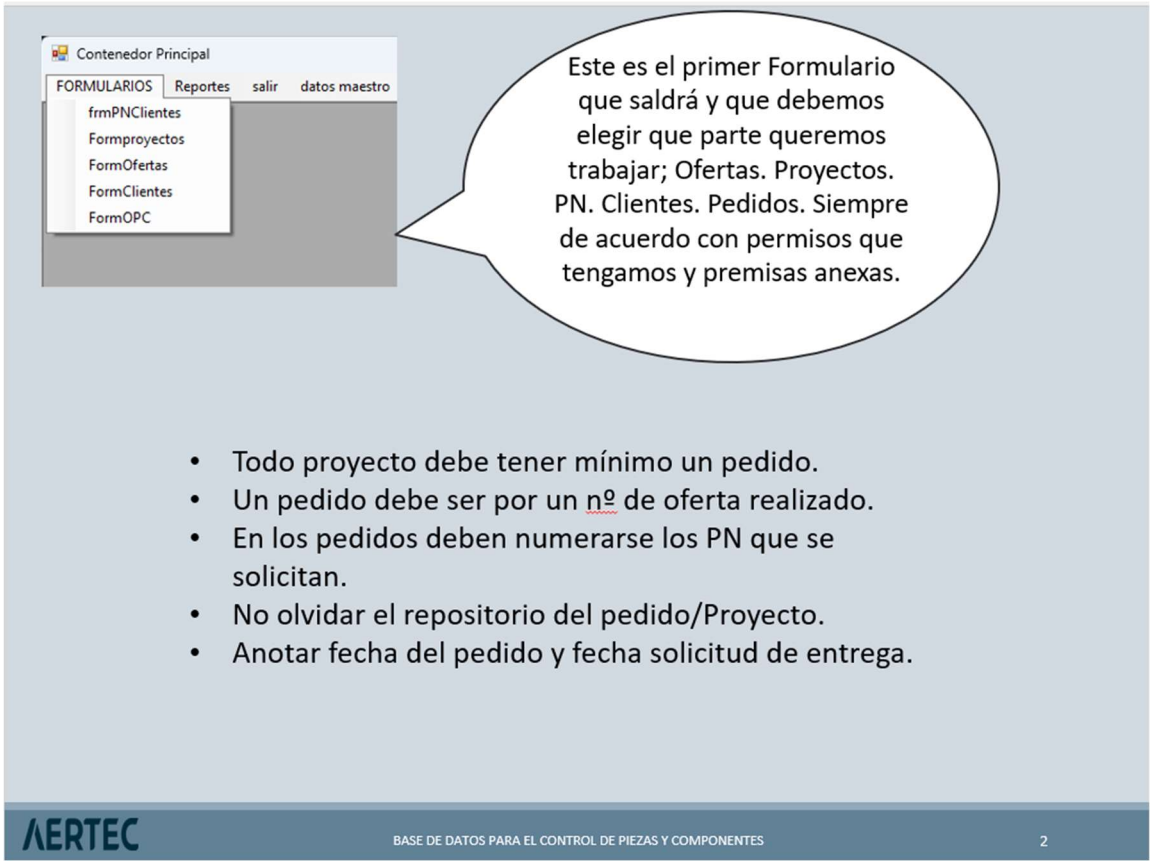
```
sing MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using MTD_CRUD_PN.DATOS;
using MTD_CRUD_PN.ENTIDADES;

namespace MTD_CRUD_PN.DATOS
{
  public class D_Usuarios
  {
    public bool VerificarUsuario(string nombreUsuario, string clave)
    {
      bool resultado = false;
      MySqlConnection conexion = new MySqlConnection();
      try
      {
        conexion= conexion.getInstancia().CrearConexión();
        MySqlCommand comando = new MySqlCommand("verificar_usuario", conexion);
        comando.CommandType = CommandType.StoredProcedure;
        comando.Parameters.Add("pNombreUsuario", MySqlDbType.VarChar).Value = nombreUsuario;
        comando.Parameters.Add("pClave", MySqlDbType.VarChar).Value = clave;
        conexion.Open();
        resultado = Convert.ToBoolean(comando.ExecuteScalar());
      }
      catch (Exception ex)
      {
        throw new Exception("Error al verificar usuario: " + ex.Message);
      }
      finally
      {
        if (conexion.State == ConnectionState.Open)
          conexion.Close();
      }
      return resultado;
    }
  }
}
```

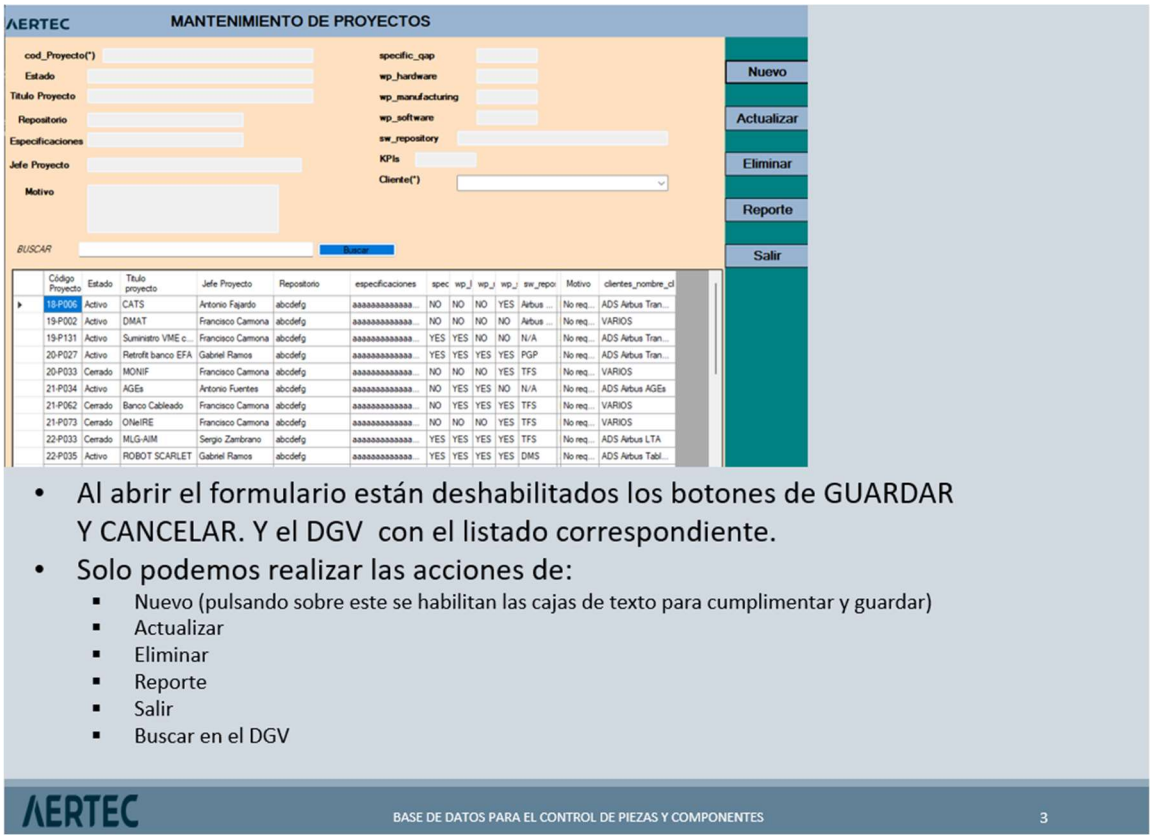
ANEXO 9 PROCEDIMIENTO ALMACENADO VALIDAR USUARIO

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `ValidarUsuarioPN`(
IN `p_nombre_usuario` VARCHAR(50),
IN `p_clave` VARCHAR(100)
)
LANGUAGE SQL
NOT DETERMINISTIC
CONTAINS SQL
SQL SECURITY DEFINER
COMMENT ''
BEGIN
    DECLARE valid INT;
    SELECT COUNT(*) INTO valid FROM usuarios
    WHERE nombre_usuario = p_nombre_usuario AND clave = p_clave AND activo = TRUE;

    IF valid > 0 THEN
        SELECT 'OK' AS Resultado;
    ELSE
        SELECT 'ERROR' AS Resultado;
    END IF;
EN
```

ANEXO 10 FORMACIÓN APLICACIÓN AERTEC

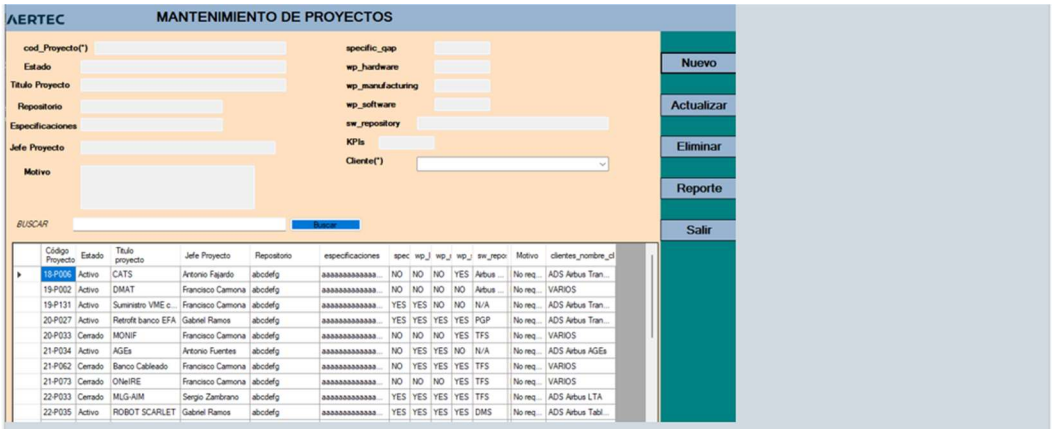






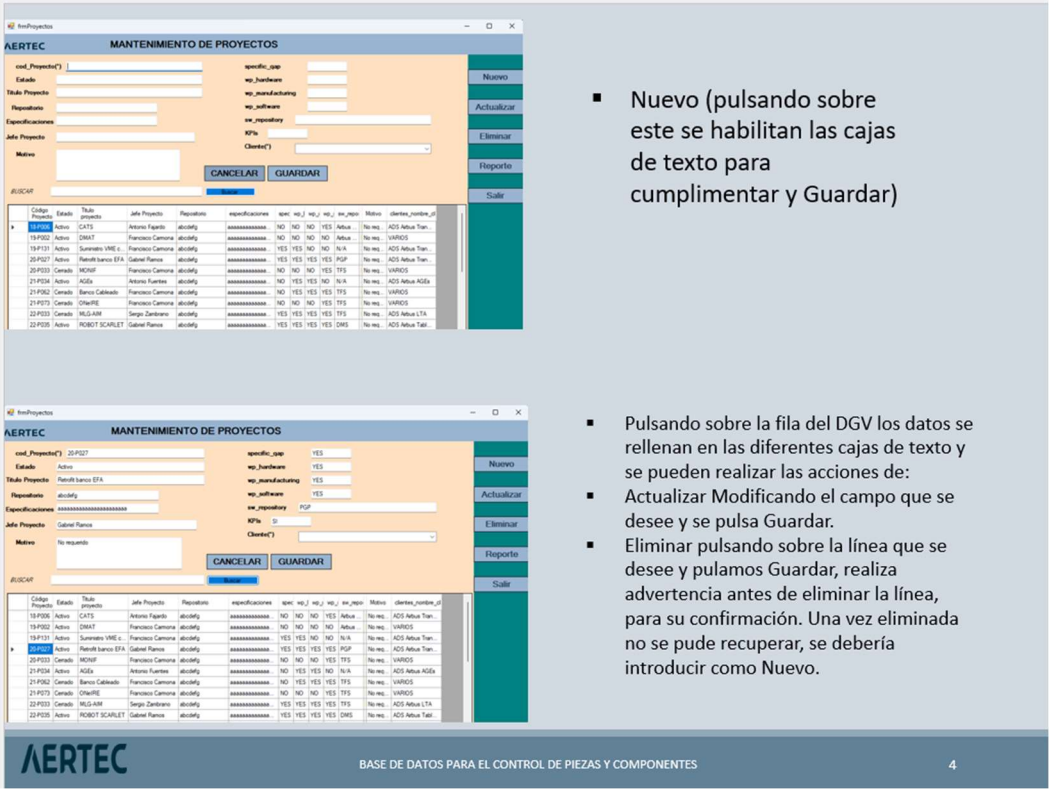
- Al abrir el formulario están deshabilitados los botones de GUARDAR Y CANCELAR. Y el DGV con el listado correspondiente.
- Solo podemos realizar las acciones de:
  - Nuevo (pulsando sobre este se habilitan las cajas de texto para cumplimentar y guardar)
  - Actualizar
  - Eliminar
  - Reporte
  - Salir
  - Buscar en el DGV

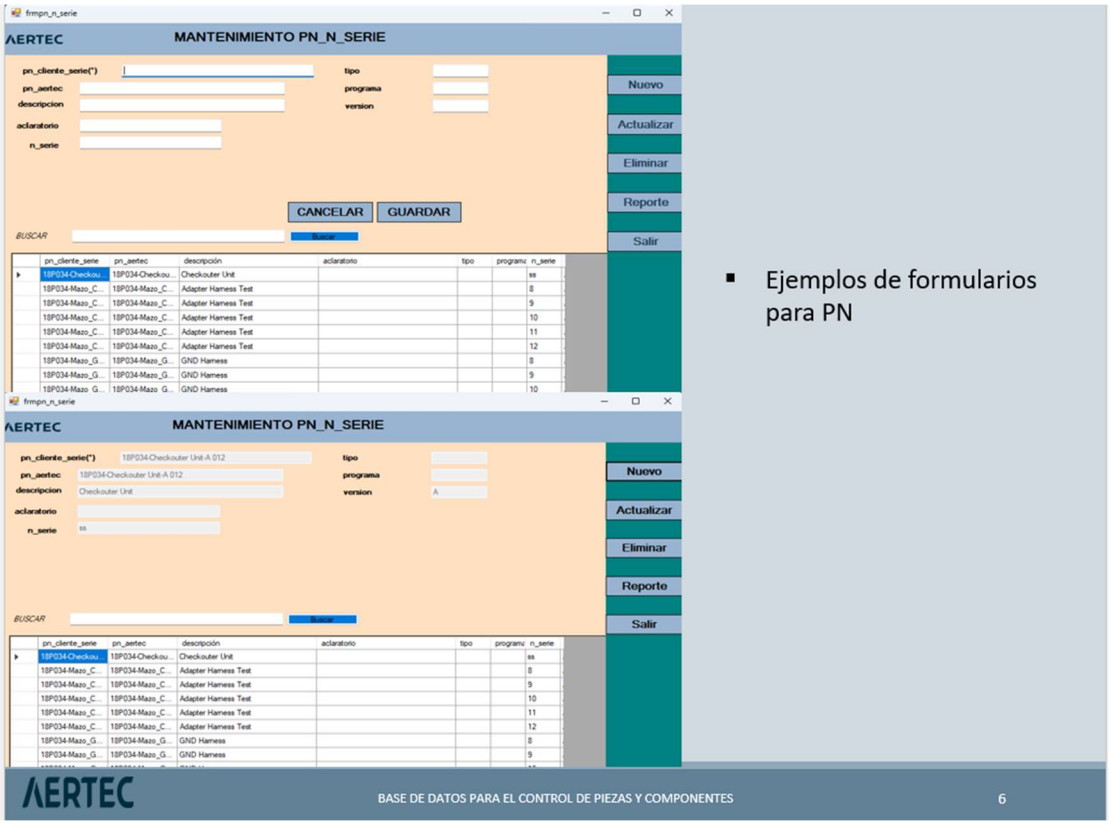
AERTEC BASE DE DATOS PARA EL CONTROL DE PIEZAS Y COMPONENTES 3

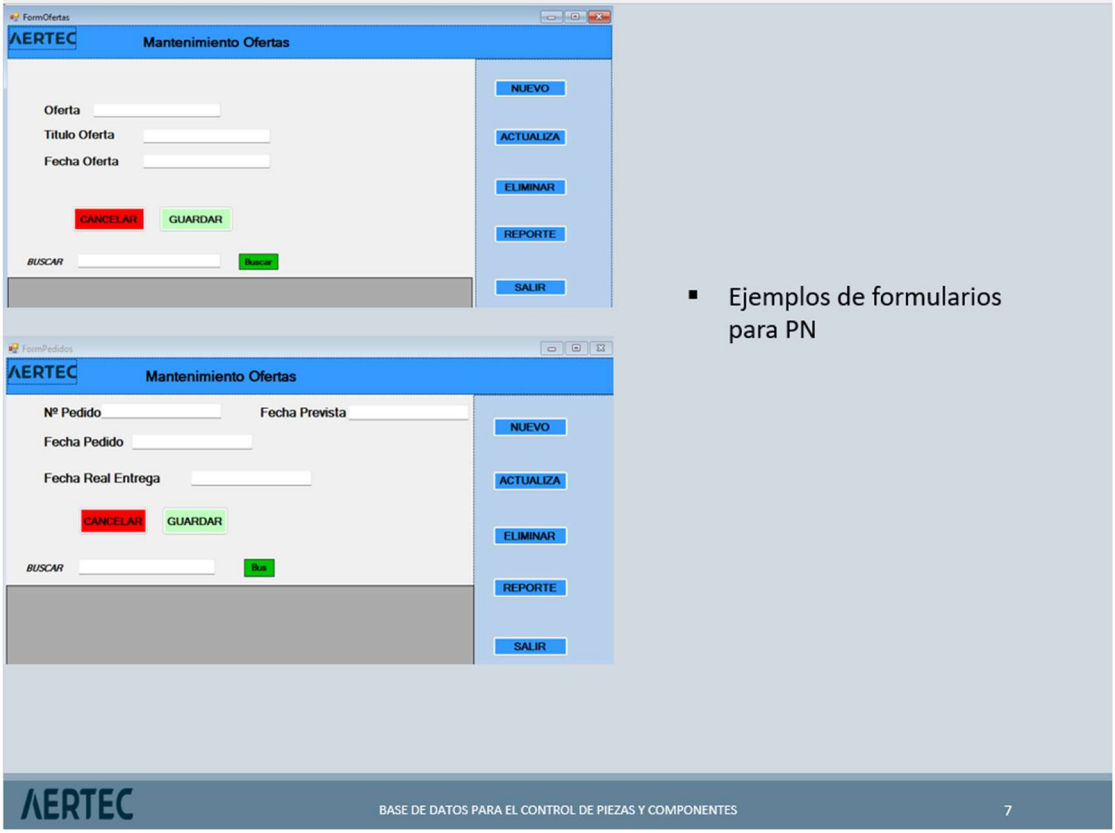


- Al abrir el formulario están deshabilitados los botones de GUARDAR Y CANCELAR. Y el DGV con el listado correspondiente.
- Solo podemos realizar las acciones de:
  - Nuevo (pulsando sobre este se habilitan las cajas de texto para cumplimentar y guardar)
  - Actualizar
  - Eliminar
  - Reporte
  - Salir
  - Buscar en el DGV

AERTEC BASE DE DATOS PARA EL CONTROL DE PIEZAS Y COMPONENTES 3







Resumen

Resumen en español

El presente trabajo detalla el desarrollo completo de una base de datos diseñada para unificar el repositorio de ofertas y proyectos, así como para inventariar los números de serie de los productos fabricados, facilitando futuros encargos de los clientes. El objetivo principal es automatizar y gestionar eficientemente los procesos productivos. Para el diseño de la base de datos se ha utilizado MariaDB, y el desarrollo se ha llevado a cabo en lenguaje de programación C#.

Además de centralizar la información, el sistema proporciona trazabilidad del producto y asegura el cumplimiento de los requerimientos técnicos solicitados por los clientes. Este enfoque no solo mejora la eficiencia operativa, sino que también garantiza un control riguroso y seguimiento continuo de los productos desde su fabricación hasta la entrega final.

Palabras clave: Trazabilidad, requerimientos técnicos, control, seguimiento, automatización, gestión de proyectos.

Abstract

Resumen en inglés

This work details the complete development of a database designed to unify the repository of offers and projects, as well as to inventory the serial numbers of the manufactured products, facilitating future orders from customers. The main objective is to automate and efficiently manage production processes. MariaDB has been used for the database design, and the development has been carried out in C# programming language.

In addition to centralizing information, the system provides product traceability and ensures compliance with the technical requirements requested by customers. This approach not only improves operational efficiency, but also ensures rigorous control and continuous monitoring of the products from manufacturing to final delivery.

Keywords: Traceability, technical requirements, control, tracking, automation, project management.