Universidad de las Ciencias Informáticas
Facultad 4

Componente para la gestión de datos de piezas normalizadas en aplicación de Diseño Asistido por Computadora.

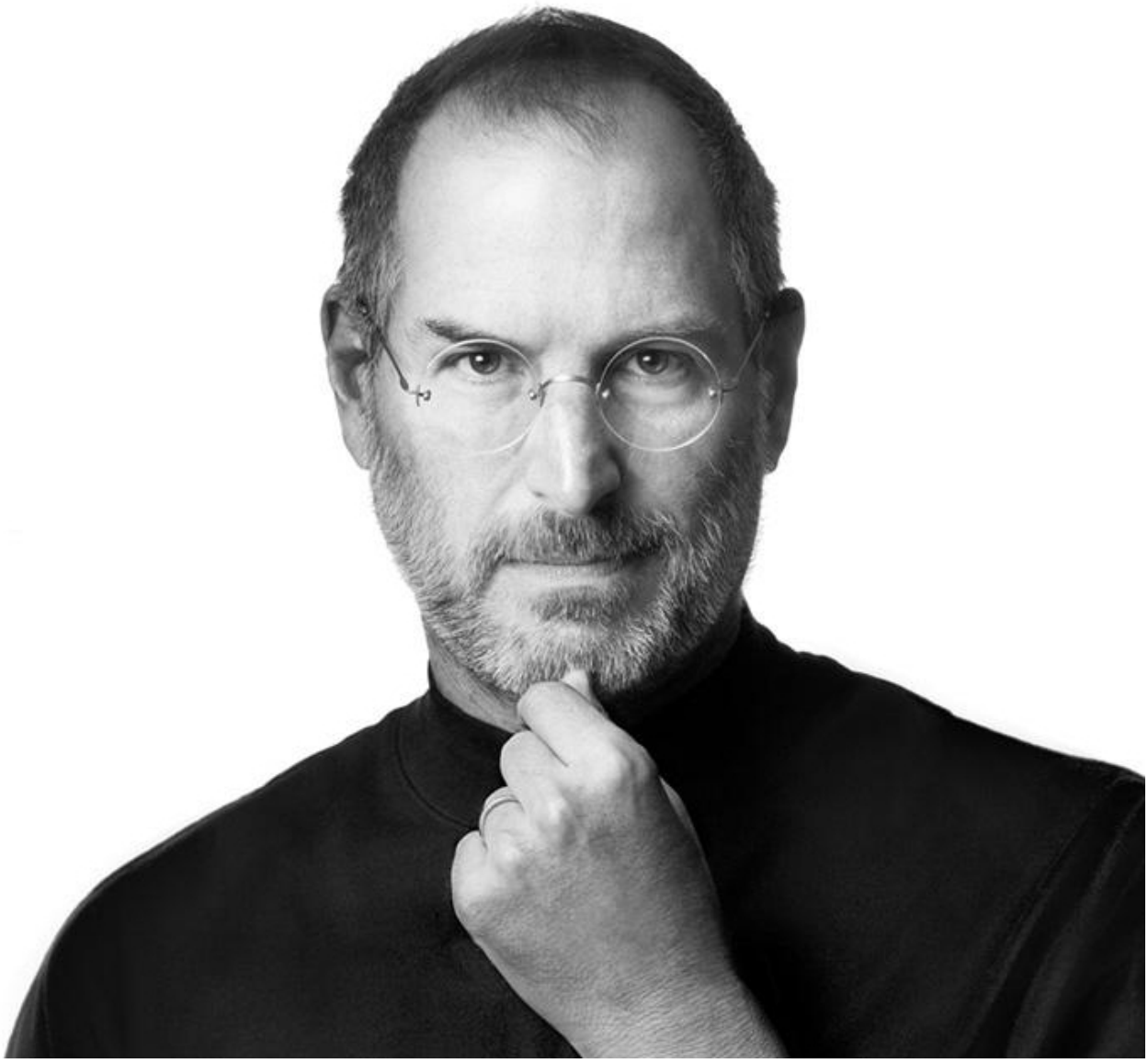
Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Arlet Apezteguía Rigueira

Tutores: Dr.C. Augusto César Rodríguez Medina

Ing. Fredy Almenares Fleitas

La Habana, 2019



La única manera de hacer un gran trabajo, es amar lo que haces. Si no lo has encontrado, sigue buscando. No te conformes.

Steve Jobs

Dedicatoria

Dedicada a mi mamá y a mi abuela por ser las personas más importantes en mi vida, mi apoyo incondicional durante la carrera y sobre todo por estar en los momentos más difíciles.

Agradecimientos

Agradezco infinitamente a mi familia por su apoyo durante estos 5 años, especialmente a mi mamá, por enseñarme que nada es imposible si se hace con amor y perseverancia, a mi abuela Gudelia por ser mi voz de aliento en todo momento y a mis padres Antonio y Ramón por los sacrificios realizados, dedicación y amor. A la persona que me ha brindado su amor y compañía siendo mi principal apoyo en la universidad mi novio Karel, también a su familia por hacerme parte de la suya y siempre estar para mí. A mis tutores Augusto y Fredy por poder contar con ellos siempre durante todo el proceso, por guiarme y alentarme a continuar sin rendirme. A todas mis amistades del apartamento por escucharme y pasar tantos momentos lindos juntos, en especial a Eliany a la que considero como parte de mi familia. A Andy por brindarme su apoyo y amistad y a todos los profesores que contribuyeron a que hoy me esté formando como Ingeniera en Ciencias Informáticas. Gracias a todas y cada una de las personas que hicieron posible que este momento fuese una realidad. A todos muchas gracias.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Arlet Apezteguía Rigueira
Autor

Dr.C. Augusto César Rodríguez Medina
Tutor

Ing. Fredy Almenares Fleitas
Tutor

Resumen

El Diseño Asistido por Computadora (CAD) estudia el uso de sistemas informáticos como herramientas de soporte en todos los procesos involucrados en el diseño y la fabricación de cualquier tipo de producto. Estos sistemas se han convertido en un requisito indispensable para la industria actual que se enfrenta a la necesidad de mejorar la calidad y acortar los tiempos de diseño y producción. Debido a que las bases de datos son uno de los principales componentes de los sistemas modernos de diseño asistido por computadora y la necesidad del grupo de investigación Soluciones Informáticas para la Ingeniería y la Industria de no contar con una herramienta que permita la gestión de datos de piezas normalizadas. Se implementó como solución a dicha situación un componente que mediante la creación y conexión de una base de datos relacional, gestiona los datos de piezas normalizadas para el sistema CAD "Ingeniero", ofreciendo la posibilidad de conexión local y remota que pueda usarse en la práctica de la ingeniería en el país. Para guiar el desarrollo de la solución se utilizó la metodología AUP en su versión UCI, lenguaje de programación C++, *framework* de desarrollo *QtCreator*, sistema gestor de base de datos PostgreSQL, entre otras herramientas. Además, se realizaron pruebas de calidad evidenciando que el componente implementado funciona correctamente y cumple con los requisitos especificados por el cliente.

Palabras clave: base de datos relacional, Diseño Asistido por Computadora, piezas normalizadas.

Índice general

Índice de figuras

Índice de tablas

Introducción

El diseño es una de las áreas más favorecidas en las empresas con la introducción de las Tecnologías de la Información y las Comunicaciones (TIC), mediante la utilización de las herramientas de Diseño Asistido por Computadora o como también se le conoce a partir de sus siglas en inglés CAD (*Computer Aided Design*). Las herramientas de Diseño Asistido por Computadora también abrieron nuevas oportunidades, como la habilitación de instrucciones de fabricación en sistemas de Planificación de Procesos Asistida por Computadora (CAPP, de *Computer Aided Process Planning*) y CAM (*Computer Aided Manufacturing*). Estas herramientas se han incrementado para satisfacer necesidades de ingeniería cada vez más complejas y con estas los métodos para transferir la información de los modelos de datos de un producto determinado (Michel, 2015).

El Diseño Asistido por Computadora permite el uso de un amplio rango de herramientas computacionales que asisten a ingenieros, arquitectos y a otros profesionales del diseño en sus actividades; es todo un sistema informático destinado a asistir al diseñador en su tarea específica. Los sistemas de Diseño Asistido por Computadora automatizan las formas tradicionales de trabajar con lápiz y papel, pero no pueden representar las reglas y las relaciones de un diseño. A medida que el hardware se vuelve más rápido y la memoria menos costosa, se adoptan tecnologías de software más sofisticadas (Kern, 1994).

Las herramientas CAD de hoy automatizan varios aspectos de la redacción, pero no nos permiten registrar los motivos de las decisiones que tomamos, las reglas y las relaciones que rigen el diseño. Un dibujo no puede transmitir estas dependencias de diseño; por eso los guardamos en nuestra cabeza. Sería un avance significativo si las herramientas de diseño basadas en computadora le permitieran registrar las relaciones de diseño que se pretende, junto con las decisiones específicas que logran estas intenciones; modelado relacional y restricciones en una base de datos.

La gestión de datos en sistemas CAD se ha realizado mediante mecanismos embebidos dedicados. Esta puede ser una buena solución para un diseñador, pero las tendencias actuales en ingeniería imponen la necesidad de compartir datos entre diversos usuarios y aplicaciones para diferentes propósitos. Se podrían introducir inconsistencias en la base de datos que son caros de eliminar. La necesidad de un diseñador y otros miembros del equipo de diseño para ver los datos de Diseño Asistido por Computadora desde diferentes perspectivas no suelen ser soportado, mucha de la información de Diseño Asistido por Computadora es interpretado y no explícito; ahí está la necesidad para que la semántica de los datos de Diseño Asistido por

Computadora sea transmitida por Sistemas de Gestión de Bases de datos (SGBD) (Kern, 1994).

Las primeras empresas y organizaciones que empezaron a usar sistemas informáticos trabajaban con programas que manejaban información almacenada en ficheros, es decir cada empresa trabajaba con sus propios datos y programas y se encargaba de su mantenimiento y gestión. Durante un tiempo esta forma de trabajo funcionó, pero con el tiempo fue aumentando la cantidad de información surgiendo problemas de integridad y duplicidad. Estos llevaron finalmente a organizar la información mediante sistemas basados en bases de datos.

Las bases de datos son componentes indispensables en la actual sociedad de la información, debido a las ventajas que presentan al organizar sistemática y eficientemente, relacionar, proteger, y administrar los datos. El origen de las mismas se da frente a la necesidad de almacenar grandes cantidades de información para su posterior consulta (Díaz, 2014).

En la actualidad existen compañías que utilizan programas de Diseño Asistido por Computadora, un ejemplo es *SolidWorks*. Este programa de diseño mecánico desarrollado por *SolidWorks Corp*, es un modelador de sólidos paramétrico, que usa el *kernel* de modelado geométrico *Parasolid*. El programa consiste en modelar piezas y extraer la información necesaria. Otro ejemplo de estos programas es el *Autodesk AutoCAD*, el cual pertenece a la empresa *Autodesk* (California, Estados Unidos), este programa permite crear dibujos técnicos detallados, usado para gráficos y siluetas en movimiento. CATIA es usada en la industria del automóvil para el diseño y desarrollo de componentes de carrocería y desarrollar edificios de gran complejidad; por ejemplo el museo de la fundación Guggenheim en Bilbao, España, es un hito arquitectónico que ejemplifica el uso de esta tecnología. Solid Edge es un programa para modelar piezas en 3D de distintos materiales, doblado de chapas, ensamblaje de conjuntos, soldadura, funciones de dibujo en plano para ingenieros, diseñadores, etc (*Introducción a los Sistemas CAD/CAM/CAE* s.f.). Autodesk Inventor se utiliza en diseño de ingeniería para producir y mejorar productos nuevos, permitiendo que las computadoras asistan en la manufactura e ingeniería (*Autodesk Inventor Professional* s.f.).

Todos los productos mencionados anteriormente contienen librerías para el modelado de piezas normalizadas o componentes normalizados, la normalización se define como el conjunto de normas que regulan todos los elementos que intervienen en las representaciones gráficas. Mediante la normalización se regulan los tamaños de la pieza que se dibuja. De esta forma cualquier persona, es capaz de interpretar los dibujos técnicos si estos se han elaborado siguiendo las normas establecidas (*Sitio Oficial ISO* s.f.).

Debido a que los sistemas de Diseño Asistido por Computadoras comerciales tienen un elevado precio de licencia para su uso; por el bloqueo tecnológico al que está sujeto Cuba, no está disponible para sus profesionales el acceso a dichas licencias a través de canales propiamente legales (Oca Montano, 2015). Cuba, a pesar de sus condiciones económicas y su lucha por desarrollarse en un mundo globalizado y en crisis, hace un gran esfuerzo por dar pasos de avance en el desarrollo de la informatización de la sociedad y la automatización de las actividades industriales y empresariales; para lograr esto se preparan profesionales a lo largo y ancho del país.

La Universidad de las Ciencias Informáticas (UCI), surgida al calor de la batalla de ideas, constituye un ejemplo de estos avances. En dicha universidad se encuentra el grupo de investigación Soluciones Informáticas para la Ingeniería y la Industria (SIPII), perteneciente al Departamento de Ciencias Básicas de la Facultad 4; como parte de sus actividades investigativas se encuentra en fase desarrollo el sistema CAD **Ingeniero**, destinado a los sectores de la industria nacional vinculados a la actividad de diseño.

Entre los componentes fundamentales de los sistemas para el Diseño Asistido por Computadora modernos, se encuentran los de bases de datos, los cuales pueden almacenar las características de componentes normalizados. Se define una Base de Datos (BD) como una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular (Carvajal y León, 2011).

En el contexto del grupo de investigación se han desarrollado módulos similares al conjunto de programas comerciales mencionados. Algunos de estos módulos necesitan cumplir con el paradigma de diseño de bases de datos; para almacenar, administrar, gestionar y usar los datos de componentes normalizados en los procesos de diseño. Uno de los problemas es que no se dispone de un componente que soporte dicho paradigma en el sistema de Diseño Asistido por Computadora que se desarrolla en el grupo de investigación. Se desconoce de la existencia de precedentes de sistemas de bases de datos para tecnologías de diseño asistido por computadora con tecnologías de código abierto que puedan usarse en la práctica de la ingeniería en el país.

La situación anteriormente descrita, unida a la necesidad del uso adecuado de herramientas de Diseño Asistido por Computadora como soporte a la mejora continua, permitieron identificar como **problema de investigación**: ¿Cómo gestionar las piezas normalizadas en el sistema de Diseño Asistido por Computadora "Ingeniero", estableciendo conexión con una base de datos relacional?

Para solucionar el problema planteado se formula como **objetivo general**: Desarrollar un componente que permita la gestión de piezas normalizadas en el sistema de Diseño Asistido por Computadora "Ingeniero", estableciendo la conexión con una base de datos relacional.

El **objeto de estudio** de la investigación es el almacenamiento de los datos de piezas normalizadas en bases de datos relacionales.

De la relación entre el problema, el objeto de estudio y el objetivo se identifica como **campo de acción**, piezas normalizadas para aplicaciones CAD.

Para dar cumplimiento al objetivo general se proponen las siguientes **tareas de investigación**:

Etapa de investigación:

- Búsqueda y revisión bibliográfica sobre el objeto de investigación, lo que incluye, el estudio del estado del arte sobre la gestión de piezas normalizadas para aplicaciones CAD, así como la asimilación de los conceptos y tecnologías requeridos para el proceso de desarrollo.

- Fundamentación de las necesidades y requerimientos de proyecto que avalan el desarrollo del sistema.
- Obtención de requisitos a partir de las fuentes disponibles (información disponible en publicaciones científico-técnicas y foros especializados colocados en Internet) e interacción con el usuario.
- Diseñar la estructura del componente en función de la arquitectura predefinida de la aplicación, lo que implica además definir los patrones de diseño con los que cumplirá, además de modelar el flujo de datos del mismo.
- Estudio de los aspectos de la metodología de software empleada en el proyecto.

Etapa de síntesis:

- Elaboración y presentación de la propuesta, obtención de conclusiones, resultados, definición de los procedimientos para el proceso de desarrollo.
- Diseño del componente para la gestión de piezas normalizadas.
- Diseño de la interfaces gráficas requeridas para el sistema.
- Implementación del componente para la gestión de piezas normalizadas con las siguientes funcionalidades:
    1. Posibilidad de instalación del sistema en un servidor remoto o en la estación de trabajo (variante "Standalone").
    2. Creación y eliminación de nuevos componentes o subcomponentes.
    3. Edición de los contenidos para actualizar o escalar el sistema.
    4. Crear contenidos para diferentes ramas de la ingeniería.
    5. Enlace o conexión con el sistema de diseño de componentes estándar.
- Integración del componente en la aplicación del proyecto.

Etapa de pruebas:

- Pruebas al sistema (de las diferentes funcionalidades y de integración).

Para resolver y dar cumplimiento al objetivo general de la investigación y a las tareas propuestas se emplean un conjunto de **métodos científicos** abordados en (Álvarez, 2018) los cuales se definen a continuación:

✓ **Métodos teóricos:**

Análisis-Síntesis: Este método permitió el análisis de la documentación de los sistemas gestores de bases de datos, así como el estudio de las técnicas, sugerencias y soluciones ya desarrolladas por otros usuarios, para poder implementar un componente estableciendo la conexión con una base de datos estable y segura.

Modelado: Este método se utilizó para realizar el modelado del sistema informático a través de los artefactos correspondientes a las fases: Modelo del Negocio, Requisitos, Análisis y Diseño e Implementación.

✓ **Métodos empíricos:**

Observación: Este método se empleó para caracterizar las soluciones, teniendo en cuenta el funcionamiento de las base de datos en los sistemas de CAD como (SolidWorks, Autodesk Inventor, etc), para establecer los requisitos con las principales funcionalidades de estos.

Experimentación: Este método se utilizó para realizar pruebas de funcionalidad y de integridad al componente verificando que funcione como espera el cliente.

Estructura del contenido:

La investigación quedó estructurada en 3 capítulos, donde se describe el trabajo realizado y los resultados obtenidos.

En el Capítulo 1 se proporciona una visión general de la naturaleza y propósito de los sistemas gestores de bases de datos, abordando los antecedentes, conceptos fundamentales, tecnologías, tendencias y lenguajes a utilizar para la implementación del componente, todo esto con el fin de alcanzar una base de conocimiento conceptual del tema a tratar.

En el Capítulo 2 se realiza la descripción y análisis de la solución propuesta mediante un levantamiento de requisitos que tributan a posibles entidades para la conformación del modelo de datos. Se realiza un modelo de la base de datos del sistema a partir de los conceptos identificados y las relaciones entre ellos, aplicándose patrones de diseño, además de la realización del diagrama de clases, el modelo conceptual y las historias de usuarios. Luego de esto se implementa el componente y se integra en la aplicación del proyecto.

En el Capítulo 3 se realiza las pruebas al sistema ofreciendo información acerca de la validación teórica así como funcional, realizada a la propuesta, con el objetivo de evaluar su diseño y comportamiento al ser gestionado mediante consultas. Se llevan a cabo pruebas de funcionalidad y de integración, garantizando que el componente cumpla con las expectativas del cliente.

CAPÍTULO 1

Fundamentación Teórica

Las bases de datos son el método preferido para el almacenamiento estructurado de datos. Desde las grandes aplicaciones, hasta los teléfonos móviles utilizan las mismas para asegurar la integridad de la información. El objetivo de este capítulo es identificar los fundamentos teóricos asociados con el empleo de los sistemas de bases de datos para gestionar la aceleración del diseño de piezas normalizadas.

1.1. Empleo de bases de datos en sistemas de Diseño Asistido por Computadora

El empleo de las bases de datos en sistemas de diseño asistido por computadora proporciona el soporte para almacenar de forma permanente la información de los diferentes objetos diseñados. Su diseño plantea una serie de problemas específicos, por la naturaleza de la información y por las constantes necesidades de cambio de la estructura, dada la naturaleza dinámica de un sistema CAD. La tendencia en los sistemas comerciales de diseño ha sido el uso del lenguaje *SQL* para realizar consultas a los datos, garantizando la seguridad de los datos mediante una base de datos relacional (*Introducción a los Sistemas CAD/CAM/CAE* s.f.).

La base de datos relacional es un conjunto de tablas, similares a las tablas de una hoja de cálculo, formadas por filas (registros) y columnas (campos). Los registros representan cada uno de los objetos descritos en la tabla y los campos los atributos de los objetos. En la base de datos relacional las tablas comparten algún campo entre ellas, estos campos son los encargados de establecer relaciones entre las tablas que permitan consultas complejas (Alonso, 2012).

La siguiente figura muestra una representación del diseño de una base de datos relacional, mediante la tabla *Bolts* y *AS 1427H-Metric*, especificando campo clave o llave primaria, campo foráneo y la relación existente entre dichas tablas.

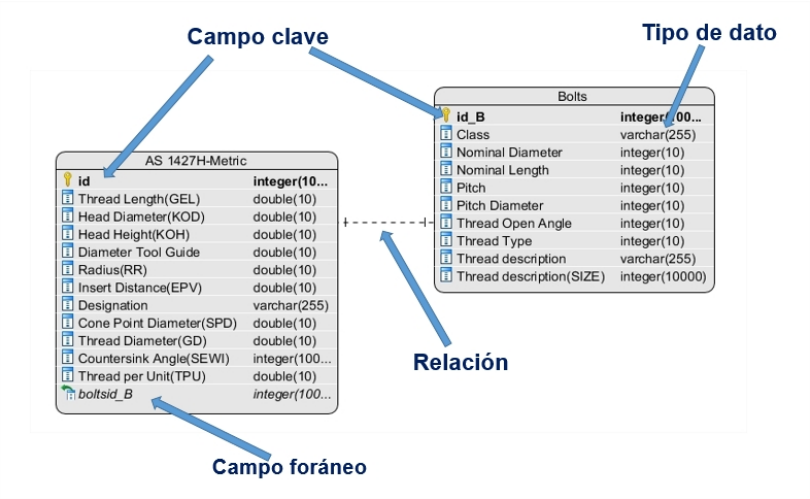


Figura 1.1. Componentes de una tabla relacional.

Según lo planteado en (*Introducción a los Sistemas CAD/CAM/CAE* s.f.) una base de datos CAD debe admitir:

- Aplicaciones de ingeniería desde diseño conceptual hasta operaciones de manufactura.
- Modificación dinámica y extensión de la base de datos y su asociatividad.
- Soporte de base de datos temporal.
- Fácil acceso.

Las bases de datos pueden clasificarse de varias formas, de acuerdo al contexto que se esté utilizando. Según la variabilidad de los datos almacenados se clasifican en bases de datos estáticas y dinámicas, las bases de datos estáticas se utilizan para almacenar datos históricos que puede ser útil para la toma de decisiones y en las dinámicas la información cambia con el tiempo, permitiendo operaciones de (insertar, modificar y eliminar) (Díaz, 2014).

Según la tendencia de acuerdo al modelo de administración de datos, las bases de datos más utilizadas han sido:

- Base de datos relacional.
- Base de datos jerárquicos.
- Base de datos de red.
- Base de datos orientada a objetos.

El modelo de base de datos relacional es utilizado en la actualidad para modelar problemas reales. Al ser propuesto en 1970 por Edgar Frank Codd, no tardó en posicionarse como un nuevo paradigma en los

modelos de base de datos. Este modelo se basa en el uso de relaciones, estas relaciones se consideran como conjuntos de datos llamados tuplas. Cada relación se muestra como una tabla compuesta por registros (filas), que representan las tuplas, y campos (las columnas). Las tuplas se identifican mediante una llave o clave primaria que son únicas en la relación, estas son necesarias para relacionar una tabla padre con una tabla hijo por medio de llave foránea. Estas bases de datos no permiten la existencia de dos tablas con el mismo nombre. El lugar y la forma en que se almacenen los datos no tienen relevancia por lo que es más fácil de entender y de utilizar para un usuario y la información puede ser recuperada o almacenada mediante consultas.

Según el autor Alejandro Gutiérrez Díaz, las ventajas que ofrece el modelo relacional son:

- Garantía de independencia de datos.
- Conectividad garantizada con los lenguajes de programación estándar.
- Favorece la normalización para ser más comprensible y aplicable.
- Compatibilidad y estandarización.
- Garantiza herramientas para evitar la duplicidad de registros a través de campos claves o llaves.
- Garantiza la integridad referencial, así al eliminar un registro elimina todos los registros relacionados dependientes.

Por su parte, el modelo jerárquico organiza los datos en una forma similar a un árbol, donde un nodo padre puede tener varios hijos y un nodo hijo solo puede tener un padre. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas. Se basan en las relaciones de cardinalidad de 1:N (uno a muchos). Las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos. Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. La mayor ventaja de un modelo jerárquico es la existencia de sistemas de manejo de BD probados que usan el modelo jerárquico como estructura básica por ejemplo *System 2000 de Intel Corporation* (Díaz, 2014).

Desventajas del modelo jerárquico:

- Las relaciones muchos a muchos pueden implantarse de una manera deficiente. Esto puede traer como consecuencia redundancia en los datos almacenados.
- Como resultado del ordenamiento jerárquico, las operaciones de inserción se vuelven complejas.
- La eliminación de padres trae como consecuencia la eliminación de hijos, como resultado de esto, los usuarios deben tener cuidado al realizar dicha operación.

Por otro lado, el modelo de base de datos de red permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico), es decir se basa en las relaciones de M:N (muchos a muchos).

Fue una gran mejora con respecto al modelo jerárquico, ya que ofrece una solución al problema de redundancia de datos.

Según el autor (Díaz, 2014) el modelo de base de datos orientados a objetos permite almacenar en la base de datos los objetos completos (estado y comportamiento). Una base de datos orientada a objetos incorpora todos los conceptos del paradigma de objetos: encapsulación, herencia y polimorfismo y permite al diseñador especificar la estructura y operaciones de objetos complejos.

- Encapsulación: Propiedad que permite ocultar los datos al resto de los objetos.
- Herencia: Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
- Polimorfismo: Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

Ventajas de las base de datos orientados a objetos (Mesa, 2011):

- Mayor capacidad de modelado.
- Ampliabilidad.
- Adecuación a las aplicaciones avanzadas de base de datos.
- Mayores prestaciones.

Desventajas de las base de datos orientados a objetos:

- Mecanismos de consulta muy primitivos, sin un estándar independiente de la plataforma aceptado.
- Imposibilidad de procedimientos almacenados, ya que los objetos solo pueden ser consultados en el cliente.
- Inmadurez en el mercado.
- Carencia de estándares.
- La optimización de consultas compromete la encapsulación.
- Su limitación suele residir en su especialización, ya que suelen estar diseñadas para un tipo particular de objetos.

Al evaluar la literatura consultada se percibe que las bases de datos NoSQL solucionan los problemas de gestión de la información almacenada en bases de datos relacionales. Estas bases de datos son sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación y no utilizan una estructura de datos en forma de tabla donde se van almacenando los datos sino que para el almacenamiento hacen uso de otros formatos como clave-valor, mapeo de columnas o grafos, entre otras (Acens, 2017). Las bases de datos clave-valor son el modelo de base de datos NoSQL más popular, en este tipo de sistema, cada

elemento se identifica por una llave única, lo que permite la recuperación de la información de forma muy rápida. Se caracterizan por ser muy eficientes tanto para las lecturas como para las escrituras. Algunos ejemplos de este tipo son *Cassandra, BigTable o HBase*. En cambio las bases de datos documentales almacenan la información como un documento, generalmente utilizando para ello una estructura simple como *JSON o XML* y donde se utiliza una clave única para cada registro. Son las bases de datos NoSQL más versátiles por ejemplo: MongoDB o CouchDB. También existen las bases de datos en grafo donde la información se representa como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se puede hacer uso de la teoría de grafos para recorrerla. Algunos ejemplos de este tipo son Neo4j, InfoGrid o Virtuoso.

A continuación algunos ejemplos de bases de datos NoSQL más utilizadas actualmente:

Cassandra es un sistema de gestión de bases de datos desarrollado por Facebook, cuyo objetivo era crear un SGBD sin fallos y que proporcione la máxima disponibilidad. Está diseñado para almacenar grandes cantidades de datos a través de diferentes nodos, repartidos entre muchos servidores, mientras que proporciona un servicio altamente disponible sin un solo punto de fallo, lo cual es esencial para un gran servicio como Facebook.

Por su parte, MongoDB es una base de datos orientada a documentos, escrita en C++. La base de datos está basada en el almacén de documentos, lo que significa que almacena valores (denominados documentos) en forma de datos codificados. La elección del formato codificado en MongoDB es *JSON*. Es muy potente, porque incluso si los datos están anidados dentro de los documentos *JSON*.

Redis es una base de datos creada por *Salvatore Sanfilippo y Pieter Noordhuis*, se trata de una base de datos del tipo clave-valor. Es como un arreglo ilimitado en memoria para almacenar datos, datos que pueden ser cadenas, conjuntos de datos o listas. Redis no permite realizar consultas, sólo se puede insertar y obtener datos, además de las operaciones comunes sobre conjuntos (diferencia, unión e inserción).

Esta tabla ofrece una breve comparación entre las funcionalidades de las bases de datos NoSQL y las SQL:

Tabla 1.1. Tabla comparativa

Características	Base de datos NoSQL	Base de datos SQL
Actuación	Alta	Baja
Confiabilidad	Pobre	Buena
Disponibilidad	Buena	Buena
Consistencia	Pobre	Buena
Almacenamiento de datos	Datos no estructurados	Datos estructurados guardados en tablas
Escalabilidad	Alta	Alta

Por lo expuesto anteriormente se determina que el modelo relacional es el que más se acopla al problema

a resolver, al entorno a trabajar y a la aplicación en cuestión. Debido a que se tiene un esquema exacto de lo que se va a almacenar en la base de datos, los joins garantizan el acceso a la información de varias tablas a la vez y estas necesitan estar relacionadas para que no existan problemas de herencia. Almacena datos estructurados que garantizan la integridad de los datos y permite que al insertar una nueva pieza herede las características de su padre. Además permite administrar datos de forma dinámica y aplica en su diseño un proceso de normalización de los datos que evita la redundancia de la información.

1.2. Sistemas de Gestión de Bases de Datos (SGBD)

Los SGBD son software que dirigen y controlan todas las gestiones que realiza las bases de datos. Un SGBD consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a los mismos. Se dedican a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Estos sistemas permiten controlar el acceso a los datos, asegurar su integridad y recuperar datos tras un fallo del sistema (Díaz, 2014).

Según el autor las funciones que deben cumplir los sistemas gestores de base de datos son:

- Abstracción de los datos: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.
- Independencia: Consiste en la posibilidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.
- Consistencia: Los casos donde no se ha logrado eliminar la redundancia y aparecen repetidos necesitan actualizarse simultáneamente.
- Seguridad: Los datos almacenados pueden llegar a tener un gran valor de acuerdo a la confidencialidad de los datos que se están utilizando. Los SGBD garantizan que estos datos se encuentren seguros frente a usuarios que no tengan el permiso para acceder a los datos.
- Integridad: Es necesario adoptar medidas para garantizar la validez de los datos almacenados. Los datos se deben proteger ante fallos de hardware, datos introducidos por usuarios descuidados.
- Respaldo: Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de los datos almacenados y de restaurar a partir de estas copias.
- Control de la concurrencia: Se debe controlar el acceso concurrente a la base de datos, que podría ocasionar inconsistencias.
- Tiempo de respuesta: Es necesario minimizar el tiempo que el SGBD se demora en dar la información y en almacenar los cambios realizados a la base de datos.

Algunos de los sistemas gestores de bases de datos más usados son:

Libres: *MySQL*, *PostgreSQL*, *SQLite*, entre otros.

Privativos: *Oracle Database*, *Microsoft SQL Sever*, *Microsoft Access*, entre otros.

A continuación se muestra algunas plataformas que soportan los SGBD (ver Figura 1.2):

	Plataforma			
SGBD	Windows	Linux	Mac OSX	BSD
Oracle	Sí	Sí	Sí	Sí
SQL Sever	Sí	No	No	No
MySQL	Sí	Sí	Sí	Sí
Postgresql	Sí	Sí	Sí	Sí

Figura 1.2. Comparación entre los SGBD.

El sistema gestor *MySQL* utiliza lenguaje SQL y es compatible con muchas plataformas diferentes, incluyendo *Microsoft Windows*. Es un SGBD de fuente abierta *open source*, lo que significa que cualquier usuario puede ser capaz de usar y modificar el software. Cuenta con un servidor de bases de datos SQL muy rápido, multiusuario y robusto. El servidor es muy rápido, fiable y fácil de usar, presenta buen rendimiento y buena velocidad a la hora de realizar consultas.(Melgarejo y Moya, s.f.).

Oracle por otro lado, es un sistema gestor de bases de datos multiusuario muy grande, desarrollado por *Oracle Corporation*. Trabaja para gestionar de manera eficiente sus recursos, una base de datos de información entre los múltiples clientes que solicitan y envían datos en la red. *Oracle* soporta todos los principales sistemas operativos para clientes y servidores.

Microsoft Access es un software de gestión de base de datos de nivel de entrada, uno de los productos de *Microsoft* más populares. La base de datos de *Access* es poderosa para proyectos a pequeña escala. Utiliza un lenguaje SQL específico (a veces denominado *Jet SQL*).

El SGBD *PostgreSQL* es un potente sistema de base de datos relacional, que ha sido desarrollado de varias formas desde la década de 1980. Es software libre y posee altos niveles de seguridad en términos generales. Presenta una gran escalabilidad y es capaz de ajustarse al número de CPU y a la cantidad de memoria que posee un sistema de forma óptima. Este SGBD soporta distintos tipos de datos, además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos, etc. Algunos de los lenguajes en los que se puede usar son: C, C++, *Java*, *Python*, entre otros. *PostgreSQL* está considerado como el SGBD de código abierto más avanzado del mundo, ya que posee características de los más potentes sistemas como *Oracle* y *MySQL* (ibíd.).

A partir del estudio de estos sistemas gestores de base de datos, analizando ventajas y desventajas de los mismos, se hace uso de *PostgreSQL* versión 9.4, ya que es multiplataforma, brinda la oportunidad de tenerlo instalado de forma local o realizar una conexión remota a través de su administrador *PgAdmin*. Su

uso garantiza la continuidad del lenguaje SQL, buen rendimiento y seguridad de la información almacenada en términos generales.

1.2.1. Modelos de diseño:

Las bases de datos deben estar respaldadas por un diseño que garantice una buena representación y fácil manipulación de los datos, es por ello que han surgido diferentes modelos de diseño. Un modelo de datos es la descripción de una base de datos, que permiten describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan esos elementos entre sí (*Información general de modelado de datos* 2010).

Según lo planteado en (ibíd.) de acuerdo al nivel de abstracción que presentan se clasifican en:

- Modelos conceptuales: Se utilizan durante la etapa de análisis de un problema dado y están orientados a representar los elementos que intervienen en ese problema y sus relaciones. Ejemplo el Modelo Entidad-Relación.
- Modelos lógicos: Son orientados a las operaciones más que a la descripción de una realidad, por ejemplo el Modelo Relacional, que cuenta con buenas características conceptuales (Normalización de bases de datos).
- Modelos físicos: Los modelos de datos físicos son estructuras de datos a bajo nivel implementadas dentro del propio manejador. Ejemplos de estos son los Árboles B+, las estructuras de Hash.

Según los modelos descritos se utiliza para el diseño de la base de datos el modelos conceptual mediante el diagrama o modelo entidad-relación (ER). Este modelo consiste en un conjunto de objetos llamados entidades y las relaciones entre ellas, representando la estructura lógica de la base de datos. Los conceptos básicos del modelo entidad-relación son entidades, relaciones y atributos. Las entidades están identificadas por atributos o propiedades, que son la unidad menor de información de los objetos que figuran. Las relaciones pueden ser de uno a uno (1:1), de uno a muchos (1:N) y de muchos a muchos (M:N), también se les conoce como cardinalidad (Silberschatz; Korth y Sudarshan, 2002).

Según (Paré; Santillán; Costa; Ginestá y Carme Martín Escofet, s.f.) la correspondencia de cardinalidades puede ser:

- Uno a uno (1:1): Una entidad de A se relaciona únicamente con una entidad en B y viceversa. La conectividad 1:1 se denota poniendo un 1 a lado y lado de la interrelación.
- Uno a muchos (1:N): Una entidad en A se relaciona con cero o muchas entidades en B. Pero una entidad en B se relaciona con una única entidad en A. La conectividad 1:N se denota poniendo un 1 en un lado de la interrelación y una N en el otro.

- Muchos a uno (N:1): Una entidad en A se relaciona exclusivamente con una entidad en B. Pero una entidad en B se puede relacionar con 0 o muchas entidades en A. vspace0.3cm
- Muchos a muchos (M:N): Una entidad en A se puede relacionar con 0 o muchas entidades en B y viceversa.

Según lo planteado en (Silberschatz; Korth y Sudarshan, 2002) las extensiones del modelo entidad-relación son:

- La generalización/especialización permite formar una nueva entidad, mediante la unión de otras entidades. El proceso inverso se denomina especialización.
- La agregación permite formar una nueva entidad, sobre la base de una relación.

Un diseño correcto de una base de datos permite tener acceso a información exacta y actualizada, y a su vez es esencial para maximizar las ventajas de la misma como es el aumento de la seguridad y la consistencia de los datos. (*Pasos para el proceso de diseño de una base de datos* s.f.).

Pasos para el proceso de diseño de las bases de datos

- Determinar la finalidad de la base de datos.
- Buscar y organizar la información necesaria: reúne todos los tipos de información que desee registrar en la base de datos.
- Dividir la información en tablas: divide los elementos de información en entidades o temas principales.
- Convertir los elementos de información en columnas: decide qué información desea almacenar en cada tabla.
- Especificar claves principales: se debe de especificar la clave principal de cada tabla. La llave principal es una columna que se utiliza para identificar cada fila, como id.
- Definir relaciones entre las tablas: permite examinar cada tabla y posibilita decidir cómo se relacionan los datos de una tabla con las demás tablas.
- Ajustar el diseño: se debe de analizar el diseño para detectar errores para comprobar si se puede obtener los resultados previstos de las tablas.
- Aplicar las reglas de normalización: aplicar reglas de normalización de los datos para comprobar si las tablas están estructuradas correctamente.

Se puede inferir que la extensión del modelo entidad-relación que se utilizará para el diseño del componente es la generalización/especialización. Garantizando la unión de dos o más conjuntos de entidades para identificar todos aquellos atributos iguales de un conjunto de entidades, con atributos semejantes y a su

vez entidades con atributos diferentes. Tomando como base las piezas normalizadas para ahorro de tiempo, dinero y esfuerzo en el desarrollo de las mismas.

1.3. Aspectos preliminares sobre las piezas normalizadas

Los sistemas comerciales de diseño, diseñan piezas o componentes normalizados, debido a que se necesita estandarizar los modelos de diseño para lograr un ahorro en tiempo, dinero y esfuerzo en el desarrollo, fabricación, comercialización y soporte de los mismos. La normalización permite la creación de normas o estándares que establecen características comunes que deben cumplir los productos y que son respetadas en diferentes partes del mundo (Cisneros, 2017).

Algunas de las piezas normalizadas que se utilizan en los sistemas CAD son elementos de fijación (*Fasteners*), los cuales deben contener variedades de tornillos (*bolts*), tuercas (*nuts*), pasadores (*pins*), remaches (*rivets*) y arandelas (*washers*).

Tornillos

Los tornillos son piezas mecánicas que se utiliza para la fijación de un objeto. Se compone de un cuerpo (caña) alargado y enroscado que se introduce en la superficie y con una cabeza que dispone de ranuras y así realizar la fuerza correspondiente para su fijación. De acuerdo a su utilidad, los tornillos presentan diferentes características, los más comunes se fabrican con metal por su resistencia. Las cabezas de los tornillos también varían, siendo de cabeza oval, plana, tipo Phillips (con ranuras en cruz) y otros.

Figura 1.3. Partes de un tornillo.

Los tornillos presentan las siguientes características:

- Diámetro exterior de la caña: Es el grosor del tornillo medido en la zona de la rosca. Se expresa en milímetros o en el sistema inglés en fracciones de pulgada.
- Tipo de rosca: Las roscas pueden ser exteriores o machos (tornillos) o bien interiores o hembras (tuercas).
- Paso de la rosca: Es la distancia que existe entre dos crestas próximas.

- La hélice de la rosca (izquierda o derecha): La dirección en los tornillos es básicamente toda a la derecha, pero algunos ejes de máquinas presentan la rosca a la izquierda.
- Material constituyente y resistencia mecánica que posee: La mayoría de los tornillos son de acero de diversas calidades y resistencia mecánica.
- Tipo de cabeza: Permite sujetar el tornillo o realizar el movimiento giratorio con ayuda de herramientas adecuadas. Ejemplo: estrella, hexagonal, cuadrada, entre otras.

Tuercas

Las tuercas son piezas que se enroscan en la rosca del tornillo para hacer la fijación o el ajuste de la pieza que queremos unir. Tiene un agujero circular en el medio que se ajusta a la rosca del tornillo. Su forma exterior puede ser diferente para cada una, pero las más utilizadas son las hexagonales con seis lados y las cuadradas con cuatro. La tuerca siempre debe coincidir con las características del tornillo al que se va a ajustar («Tuercas y Tornillos» 2018).

Figura 1.4. Tipos de tuercas.

Existen diferentes tipos de tuercas como son:

- Hexagonal: Tiene seis caras con forma hexagonal.
- Cuadrada: Tiene cuatro caras con forma cuadrada.
- Tuerca ciega: Este tipo de tuerca presenta orificio de entrada y no de salida. Normalmente su uso es de tipo decorativo.
- Tuerca con arandela a presión: Esta tuerca es una evolución de la tuerca hexagonal, incorporando una arandela fija en uno de sus planos horizontales, la cual tiene estrías que impiden que se afloje el tornillo.
- Tuerca de seguridad: Este tipo de tuerca presenta la particularidad de incorporar un aro de nylon en uno de sus planos horizontales y su función es el bloqueo del tornillo.

- Tuerca mariposa: Su uso está vinculado a la necesidad de apriete y afloje rápido, ya que se suele hacer con la mano.
- Tuerca almenada: También conocida como tuerca de castillo, este nombre se le da debido a que su forma asemeja un castillo medieval.
- Tuerca de cabeza moleteada: Para tuercas que se enroscan a mano.
- Tuerca ranurada: Este tipo de tuerca al ser colocada, sus ranuras se agarran al material para evitar la rotación o el giro de la tuerca. El resultado es una solución de roscado altamente segura y resistente.

Pasadores

Los pasadores son piezas de acero de forma cilíndrica o cónica, cuyos extremos facilitan su introducción en un orificio común a dos o más piezas, provocando su inmovilización (pasador de fijación), o asegurando la posición relativa entre las piezas (pasador de posición) (López, 2017).

Existen diferentes tipos de pasadores entre los que se encuentran:

- El pasador cilíndrico se emplea como elemento de fijación entre dos o más piezas.
- El pasador cónico se emplea para asegurar la posición de elementos mecánicos facilitando el centrado de las piezas.
- El pasador ajustado con cabeza es un elemento de unión empleado en articulaciones que tienen habitualmente juego en el cojinete.
- El pasador estriados se utilizan en perforaciones sencillas.
- El pasador de aletas al introducirse en su lugar se doblan en sentido opuesto sus extremos produciendo su fijación.
- El pasador elástico es un cilindro hueco y tiene una ranura de un extremo a otro, para facilitar su introducción.

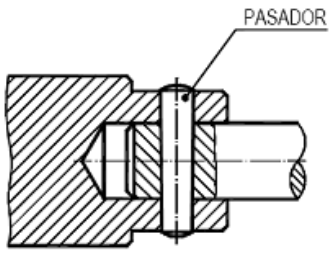


Figura 1.5. Pasador cónico.

Arandelas

Las arandelas son piezas planas en forma de disco con un agujero en el centro, usada para las uniones de tuerca y tornillo. La arandela se coloca debajo de la tuerca, y su función principal es evitar que la tuerca gire sobre el material que aprieta, evitando que éste se deforme. El agujero interior está en dependencia del tornillo al cual están destinadas (Carrasco, 2018).

Existen diferentes tipos de arandelas entre ellas:

- Las arandelas planas de hierro se utilizan para tornillos, incluso para tornillos de cabeza hexagonal. Dependiendo de su función, pueden ser de redondas, cuadrada, etc.
- Las arandelas dentadas se identifican porque son negras y con muchos dientes. Se utilizan para disminuir la fricción y así prevenir que no se afloje la unión.
- Las arandelas cónicas son más grueso hacia el centro, lo que garantiza un mayor apriete de la tuerca sin deformarse.
- Las arandelas de presión son un anillo no cerrado, sus extremos terminan en bordes filosos y una produce agarre en la pieza sostenida, y la otra en la tuerca, evitando que se afloje.

Figura 1.6. Arandelas.

Remaches

Los remaches son sistemas de fijación, cuya finalidad es parecida a la de un tornillo, permite unir de forma permanente dos o más elementos de igual o distinto material. Un remache tiene tres partes fundamentales: cuerpo del remache de forma cilíndrica, cabeza en forma de casco esférico, cuyo diámetro es mayor al cuerpo del remache y el mandril, esta última parte es la que se inserta en la remachadora («Tipos de remaches para uniones perfectas» s.f.).

Tipos de remaches:

- Remache de cabeza plana: su resistencia a la fuerza es mayor que los remaches de cabeza semicircular.
- Remache hueco: el extremo es hueco para ser conformado con un instrumento puntiagudo.
- Remache de dos piezas: una de las piezas tiene un agujero central, donde se introduce la otra pieza.
- Remache ciego o pop: es un remache con un agujero interior, en el que viene introducido un clavo con cabeza. La diferencia con los remaches comunes, es que el remache pop puede introducirse por un solo extremo sujetándolo por el clavo y colocando la cabeza en la perforación existente entre las dos superficies a unir (placas o láminas).
- Remache de cabeza semicircular: su cabeza forma una especie de colina y es resistente.

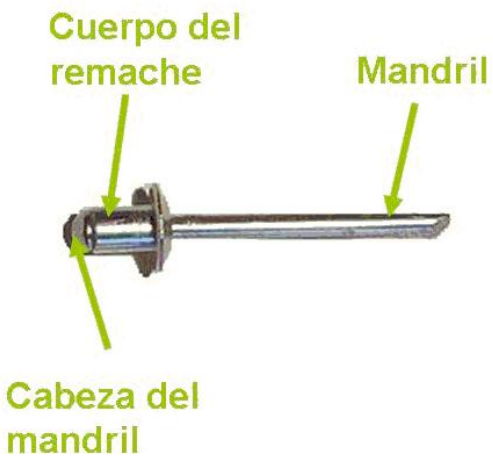


Figura 1.7. Remache ciego con mandril de estiramiento.

Las piezas normalizadas anteriormente descritas, se utilizan en el desarrollo de la base de datos, debido a que es necesario almacenar de forma permanente la información inherente a los mismos, teniéndose en cuenta las normas internacionales existentes. Tomando como base para la aceleración del diseño posteriormente a partir de esos datos almacenados.

1.4. Normas internacionales que rigen las piezas normalizadas

La aceleración del diseño de piezas normalizadas en sistemas comerciales de diseño asistido por computadora utiliza piezas estándar en la construcción de herramientas, fabricación de moldes e ingeniería mecánica. La normalización es la aprobación de normas que se establecen para garantizar el funcionamiento de elementos construidos de forma independiente. Un estándar o norma es un documento que provee requerimientos, especificaciones, características que pueden ser utilizadas para garantizar que los materiales, productos, procesos o servicios están aptos para su propósito (*Sitio Oficial ISO* s.f.).

Según la ISO (Organización Internacional de Normalización) la normalización es la actividad que tiene por objeto establecer, ante problemas reales o potenciales, disposiciones destinadas a usos comunes y repetidos, con el fin de obtener un nivel de ordenamiento óptimo en un contexto dado, que puede ser tecnológico, político o económico. La normalización se refleja en forma de documentos técnicos llamados especificaciones, legislaciones y normas. La normalización es el conjunto de indicaciones generales y de reglas que establecieron los países industrializados para favorecer el comercio y la fabricación de todo tipo de bienes y servicios. (*Sitio Oficial ISO* s.f.).

Algunas normas internacionales utilizadas en sistemas CAD para piezas normalizadas son la norma americana (ANSI), norma australiana (AS), norma británica (BSI), norma taiwanesa (CNS), norma checa (CSN), norma alemana (DIN), norma china (GB), norma india (IS), norma internacional (ISO), norma japonesa (JIS), norma polaca (PS), norma sueca (SS), norma italiana (UNI). Según lo planteado en (*Beneficios de la normalización* s.f.) la utilización de estas normas ofrece varios beneficios para las empresas y la sociedad, por ejemplo:

- Aseguran la competencia leal y brindan estímulos para mejorar la calidad.
- Ayudan a optimizar las operaciones, a disminuir costos, a incrementar la satisfacción de los clientes y mejoran en general la productividad y la competitividad.
- Las normas evitan barreras comerciales innecesarias y facilitan el acceso a los mercados mundiales.
- Los consumidores pueden tener la confianza que los productos y servicios son seguros, confiables y de calidad adecuada.

1.5. Manipulación de piezas normalizadas en sistemas CAD

Los diferentes sistemas comerciales manipulan desde su surgimiento los datos de piezas normalizadas por ejemplo en los siguientes párrafos se abordará más a fondo de la realización de este proceso.

Por ejemplo *Solidworks* es un programa de diseño asistido por computadora para modelado mecánico, desarrollado en la actualidad por *SolidWorks Corp*. Presenta un gestor de diseño *Feature Manager* que facilita la modificación rápida de operaciones tridimensionales de operación sin tener que rehacer los diseños ya plasmados en sus documentos asociados. Utiliza *Microsoft SQL Server* y proporciona un conjunto intuitivo de configurables herramientas para administrar sus archivos. Este programa permitir que toda su empresa comparta datos fácilmente mediante servidor remoto para colaborar en los diseños utilizando *Microsoft Excel* para insertar tablas de diseño en el documento de pieza. Una tabla de diseño permite construir varias configuraciones distintas de una pieza aplicando los valores de la tabla a las cotas de la pieza, se selecciona «creación automática», y se muestra una ventana con todos los parámetros configurables de ese archivo. Automáticamente aparece una hoja de cálculo, la primera columna corresponde a las configuraciones de cada pieza, el resto son los parámetros escogidos. Este proceso garantiza que se recupere fácilmente versiones anteriores de piezas, ensamblajes y dibujos mientras se trabajaba (Grado, 2015).

Por su parte, *Autodesk Inventor* permite a los fabricantes ir más allá de un simple modelo 3D, es desarrollado por la empresa de software *Autodesk*. Garantiza a los ingenieros y diseñadores crear prototipos digitales, gracias a un conjunto de herramientas para diseño mecánico en 3D, análisis, visualización y documentación. Con Inventor los ingenieros pueden integrar los planos 2D y los modelos 3D en un único modelo digital, creando representaciones digitales del producto final (*Autodesk Inventor Professional* s.f.). En su interfaz existe un centro de contenido, éste centro guarda toda la información referente a las piezas normalizadas. El centro de contenido se visualiza como un árbol de dependencia donde se encuentran los *Fasteners*: tornillos, tuercas, pasadores, arandelas, remaches y otros. Estas piezas se rigen por diferentes normas que se almacenan en forma de tabla. Inventor cuenta con aproximadamente 4000 tablas reportadas en tres hojas excel, cada tabla representa una norma y sus filas representan un conjunto de piezas que se corresponden a la misma. Cualquiera de estas piezas normalizadas puede ser visualizada en el área de trabajo principal del módulo *Assembly*, siendo sus parámetros definidos por el centro de contenidos bajo cierta norma, listo para ser utilizado como componente básico de un ensamble. Estas tablas están organizadas según diferentes categorías, cada una tiene múltiples tuplas por lo que la cantidad de información a gestionar es considerable. Se ha visto como tendencia, no solo en sistemas CAD, el apoyarse en sistemas gestores de bases de datos para lidiar con grandes volúmenes de información.

SQLServer es utilizado también por *Solid Edge* garantizando el almacenamiento de la información de forma segura. Se ofrece a los fabricantes que tengan dificultades para trabajar con volúmenes de archivos CAD que crecen rápidamente. La configuración es rápida, basta con colocar un índice en las carpetas donde se almacenan los archivos. En la administración de datos se utiliza tecnología de indexación para almacenar información acerca de sus archivos. No es necesario instalar ni mantener ningún software de bases de datos, solo se colocan índices en las carpetas donde se almacenan los archivos. *Solid Edge* ofrece soluciones de almacenamiento e intercambio de archivos utilizando opciones de almacenamiento local, en red y en la nube que satisfacen las necesidades de usuarios individuales y de grandes fabricantes. También mantiene sus archivos de forma local, ofreciendo un rendimiento óptimo y la flexibilidad de trabajar sin conexión («Funciones de gestión de datos integradas en Solid Edge» 2017).
Solid Edge usa la configuración del archivo *Standard.ini*. El valor predeterminado en el archivo Standard.ini es “Auto”, es decir las plantillas se establecerán según el formato de región e idioma del usuario actual mediante las normas internacionales («Solid Edge» 2015)

A raíz de lo expuesto anteriormente se puede apreciar que los sistemas comerciales de diseño según la tendencia utilizan lenguaje SQL. Almacenando la información de las piezas en hojas de excel, para una mejor organización de los datos y acelerar el proceso de diseño siguiendo los parámetros guardados en la base de datos. Además se reporta que utilizan servidor remoto para interactuar y colaborar desde diferentes ubicaciones.

1.6. Metodología de desarrollo de software

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce. Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable (Sánchez, 2015).

Se escoge AUP versión UCI, porque la misma es una variación de la metodología AUP, y se adapta al ciclo de vida definido para la actividad productiva de la UCI. Por lo tanto se decide escoger una metodología para ser adaptada a lo que ya la Universidad ha estado proponiendo como ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introducir la menor cantidad de cambios posibles (ibíd.).

Esta metodología de las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre (ibíd.).

Descripción de las disciplinas

La metodología AUP propone 7 disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis y diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMIDEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (ibíd.).

La versión de AUP define cuatro escenarios en los que se puede ubicar el desarrollo de una aplicación de acuerdo a sus características. El desarrollo del componente estará guiado por el escenario No 4 que aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente está siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una Historias de usuario (HU) no debe poseer demasiada información (ibíd.).

Teniendo en cuenta la política de desarrollo de software de la universidad, se define como metodología a emplear la AUP-UCI, en el escenario No 4, debido a la necesidad de que el cliente esté presente en el desarrollo de la aplicación para una mejor realización de la misma.

1.7. Conclusiones del capítulo

- El estudio del estado del arte a partir de las fuentes consultadas permitió inferir que los sistemas de diseño asistido por computadora según la tendencia emplean lenguaje SQL, debido a que se reporta su uso en *Solidworks*, *Autodesk Inventor*, entre otros.
- La tendencia de que las normas para piezas normalizadas se almacenen en tablas, sugiere que se deba utilizar un sistema gestor de base de datos que opere con los datos de la misma manera, lo que resulta factible con bases de datos relacionales.
- En los sistemas comerciales se encuentran presentes las variantes de instalar servidor local o remoto, para adaptar a la instalación del sistema a las necesidades del diseño colaborativo.

CAPÍTULO 2

Descripción de la propuesta de solución

Tomando como base las inferencias de información del primer capítulo y la evaluación de la disponibilidad de los sistemas que existen, se identifican los requerimientos de software y los artefactos generados para darle solución al problema planteado en la investigación.

2.1. Descripción de la propuesta de solución

Para el desarrollo de la propuesta de solución se utilizaron un conjunto de herramientas que garantizan que el componente funcione adecuadamente por ejemplo: La herramienta utilizada para la administración de bases de datos fue *PgAdmin III*, ya que brinda la posibilidad de que al realizarse alguna modificación en un objeto, escribe la sentencia SQL correspondiente, también incorpora funcionalidades para realizar consultas, examinar su ejecución y trabajar con los datos empleando bases de datos relacionales. Se elige el uso del lenguaje de programación C++ versión 11 y *framework* de desarrollo *QTCreator* versión 5.9.5, porque son herramientas de código abierto y fueron de útil acceso para el desarrollo del componente. Como herramienta de modelado se utilizó *Visual Paradigm* versión 8.0, facilitando la creación de artefactos ingenieriles guiado por la metodología escogida. Además de *RapidMiner* version 9.0 para la extracción, transformación y carga de la base de datos relacional.

A partir de lo anteriormente descrito, se propone el desarrollo de un componente que gestione piezas normalizadas mediante de la conexión con una base de datos relacional con la finalidad de brindar la posibilidad de su utilización a las aplicaciones de diseño asistido por computadora. Este componente se basa en una *Application Programming Interface* (API, por sus siglas en inglés) que posibilita la gestión de datos en una base de datos, la cual será utilizada por la sección *Standard* dentro de la sección de aceleradores de diseño *Accelerators*. Al activarse el botón *DataBaseManager* se muestra una ventana con diferentes opciones que permite establecer la conexión con la base de datos ya creada con el nombre BaseDatos (posee 300 tablas). Antes de ser establecida la conexión el usuario debe registrarse por Admin o Invitado, de acuerdo al rol establecido se le facilitará las acciones que puede realizar en el sistema. En el caso de que el usua-

rio en la opción del host decida insertar una dirección ip remota debe configurar en el servidor de base de datos *PostgreSQL* el fichero *postgresql.conf*, en este fichero podemos cambiar todos los parámetros de configuración que afectan al funcionamiento de *PostgreSQL* en nuestra máquina (listen_addresses = *) y el fichero *pg_hba.conf*, ya que permite controlar el acceso al servidor de base de datos (se adiciona uno nuevo insertando type = host, database = all, user = all, ip Address = 0.0.0.0/0 y method = md5).

- Opción *Host*: el usuario puede insertar un host local o remoto.
- Opción *DataBase name*: nombre de la base de datos.
- Opción *username*: nombre del usuario (Admin o Invitado).
- Opción *password*: contraseña asociada al rol establecido.
- Opción *Search DataBase*: contiene las base de datos disponibles, son reflejadas en la opción DataBase name.
- Opción *Test Connection*: permite establecer o no la conexión con la base de datos.
- Opción *Create DataBase*: permite crear una nueva base de datos.
- Opción *Remove DataBase*: permite eliminar una base de datos ya creada.

Al realizar la configuración de forma correcta se presiona el botón aceptar y aparece la *Application Programming Interface* con el nombre *Manager*, que permita al usuario acceder a la base de datos desde la aplicación, sin tener que utilizar otro programa. La interfaz permite el contacto directo con los datos de la base de datos brindándole la posibilidad al usuario de agregar, eliminar, renombrar filas y columnas además de exportar en diferentes formatos. La figura muestra la posición de la base de datos dentro de la aplicación Ingeniero, la base de datos brinda conexión a diferentes módulos y contiene piezas normalizadas (ver Figuras 2.1, 3.6).

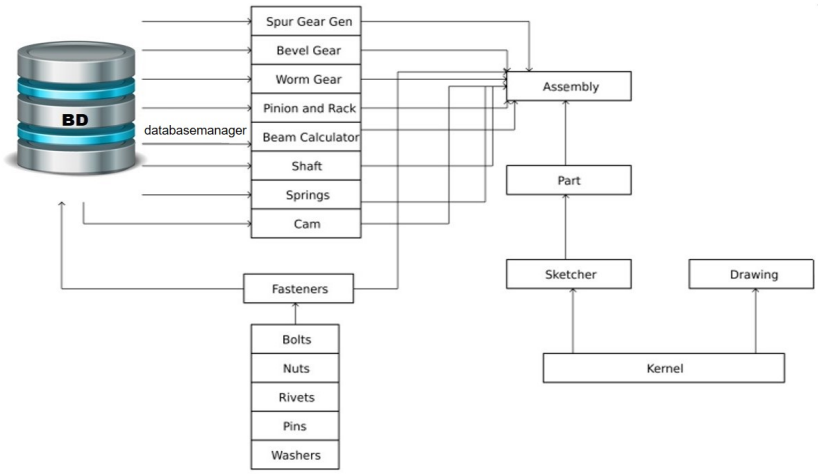


Figura 2.1. Esquema de la aplicación.

2.1.1. Estructura para la base de datos relacional

La figura muestra la transformación realizada en la herramienta *Rapidminer* para obtener como resultado la base de datos relacional. Primeramente se carga el archivo excel que contiene los datos normalizados, se estructura en tablas heredando elementos comunes de entidades padres en este caso *Fasteners* y *bolts* y luego una nueva tabla con los atributos propios de la pieza. Al finalizar esta distribución de atributos se procede al llenado de tablas mediante la conexión con la base de datos 2.3.

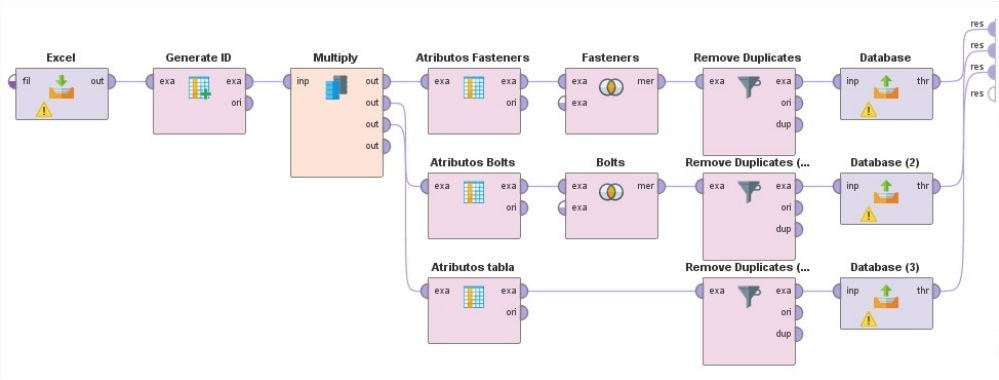


Figura 2.2. Transformación de la base de datos.

2.2. Especificación de requisitos del software

Según (Sommerville, 2005) "Un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema."Los requisitos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema determinado. Los requisitos pueden ser clasificados en requisitos del usuario y requisitos del sistema (ibíd.).

Los requisitos funcionales (RF) permiten conocer explícitamente las funcionalidades que va a presentar el sistema a desarrollar, son declaraciones de los servicios que debe proporcionar el sistema, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse en situaciones específicas (ibíd.).

El encuentro con el cliente resultó en la definición de 14 requisitos funcionales. A los requisitos se les asignó una prioridad y complejidad teniendo en cuenta la importancia establecida por el cliente a partir de sus necesidades y la dificultad con la que se realiza su implementación.

Requisitos funcionales

- RF 1. Adicionar tablas a la base de datos.

- RF 2. Adicionar columnas a la base de datos.
- RF 3. Editar nombre de las tablas.
- RF 4. Editar nombre de las columnas.
- RF 5. Eliminar tablas y/o columnas de la base de datos.
- RF 6. Realizar la comunicación desde las interfaces de la aplicación.
- RF 7. Implementar las funcionalidades necesarias en el servidor (local o remoto) de base de datos para permitir la comunicación con la aplicación.
- RF 8. Insertar las siguientes colecciones de componentes normalizados: elementos de fijación (*Fasteners*), la cual debe contener las diferentes variedades como tornillos (*bolts*), tuercas (*nuts*), pasadores (*pins*), remaches (*rivets*) y arandelas (*washers*).
- RF 9. Insertar en las tablas columnas y filas deben tener en el caso de las primeras las características del componente y en las filas las diferentes combinaciones.
- RF 10. Clasificar los datos de acuerdo a diferentes normas *(ANSI, BSI, GB, GOST, ISO, JIS, DIN, PARKER, AS, IS, KS, MIL, PEM, SKF, Torrington, Truarc, Unistruct)*.
- RF 11. Exportar un contenido seleccionado en formatos de hoja de cálculo (xls, csv, ods, etc) o LibreOfficeBase (odf, Microsoft Access Database si es posible).
- RF 12. Crear la base de datos desde la aplicación por los roles definidos para su acceso.
- RF 13. Eliminar la base de datos desde la aplicación por los roles definidos para su acceso.
- RF 14. Realizar las consultas, reportes y formularios.

Los requisitos no funcionales (RnF) son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones de la respuesta en el tiempo y del proceso de desarrollo y almacenamiento. Se conocen como un conjunto de características de calidad, que es necesario tener en cuenta al diseñar e implementar el software (Sommerville, 2005).

Requisitos no funcionales

- RnF 1. Software: El sistema debe de funcionar sobre cualquier distribución de sistema operativo Linux y/o Windows.
- RnF 2. Restricciones de diseño e implementación: La base de datos tiene que ser elaborada empleando el lenguaje SQL y debe ser una base de datos relacional.
- RnF 2.1. El sistema debe utilizar como gestor de base de datos PostgreSQL 9.4.
- RnF 2.2. El sistema debe utilizar como lenguaje de programación C++ y framework de desarrollo QT

Creator 4.5.2.

- RnF 2.3. El sistema debe utilizar como herramienta de modelado para los artefactos ingenieriles Visual Paradigm 8.0.
- RnF 3. Seguridad: Se empleará al componente autenticación mediante roles.

2.3. Historias de usuario

Las historias de usuario (HU) definen lo que se debe construir en el proyecto de software, tienen una prioridad asociada establecida por el cliente, esta prioridad depende de cuán importante es para el resultado final y su tiempo será estimado por los desarrolladores. El cliente describe brevemente las características funcionales o no funcionales que el sistema debe presentar. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento pueden eliminarse, reemplazarse por otras más específicas, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en poco tiempo (Penadés, 2013).

La HU presenta la siguiente estructura:

- Número: A cada HU se le asigna un número para facilitar su identificación.
- Nombre: Nombre descriptivo de la HU.
- Prioridad: Grado de prioridad que se le asigna a la HU en dependencia de las necesidades del cliente, (Alta, Media o Baja).
- Complejidad: Grado de complejidad que se le asigna a la HU luego de ser analizada. (Alta, Media o Baja).
- Estimación: Unidades de tiempo estimadas por el equipo de desarrollo para darle cumplimiento a la HU.
- Iteración: Número de la iteración en la cual será implementada la HU.
- Descripción: Descripción simple sobre lo que debe hacer la funcionalidad en cuestión.
- Información Adicional: Breve información que ayude a comprender algún dato no especificado antes.

Tabla 2.1. Historia de Usuario RF №1.

Número: 1	Nombre del requisito: Adicionar tablas a la base de datos
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 168h
Riesgo en Desarrollo: N/A	Tiempo Real: 1 semana
Descripción: **1- Objetivo:** Adicionar tablas a la base de datos.	

2- Acciones para lograr el objetivo (precondiciones y datos):
Para adicionar las tablas a la base de datos se debe tener en cuenta los siguientes datos:
- Nombre de la tabla.

3- Flujo de la acción a realizar:
- El usuario selecciona la opción Add Table e introduce los datos necesarios.
- Si se selecciona el botón Cancel se cierra la ventana.

Observaciones: El usuario debe tener una copia de respaldo y ser personal autorizado para acceder a la aplicación.

Prototipo de interfaz:

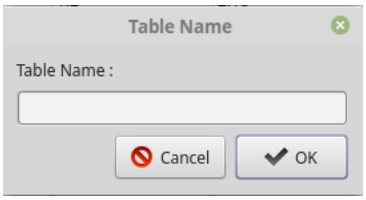


Tabla 2.2. Historia de Usuario RF №2.

Número: 2	Nombre del requisito: Adicionar columnas a la base de datos
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 168h
Riesgo en Desarrollo: N/A	Tiempo Real: 1 semana

Descripción:

1- Objetivo:
Adicionar columnas a la base de datos.

2- Acciones para lograr el objetivo (precondiciones y datos):
Para adicionar columnas a la base de datos se debe tener en cuenta los siguientes datos:
- Nombre de la tabla y tipo de dato de la misma.

3- Flujo de la acción a realizar:
- El usuario selecciona la opción Add Column e introduce los datos necesarios.
- Si se selecciona el botón Cancel se cierra la ventana.

Observaciones: El usuario debe tener una copia de respaldo y ser personal autorizado para acceder a la aplicación.

Prototipo de interfaz:

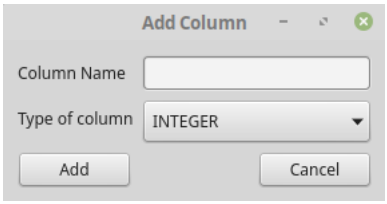


Tabla 2.3. Historia de Usuario RF №3.

Número: 3	Nombre del requisito: Editar nombre de las tablas.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 168h
Riesgo en Desarrollo: N/A	Tiempo Real: 1 semana
Descripción: **1- Objetivo:** Editar nombre de las tablas en la base de datos. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para editar el nombre de las tablas se debe tener en cuenta los siguientes datos: - Nombre de la tabla. **3- Flujo de la acción a realizar:** - El usuario selecciona la opción Rename Table e introduce los datos necesarios. - Si se selecciona el botón Cancel se cierra la ventana.	
Observaciones: El usuario debe tener una copia de respaldo y ser personal autorizado para acceder a la aplicación.	
Prototipo de interfaz: 	

Tabla 2.4. Historia de Usuario RF №4.

Número: 4	Nombre del requisito: Editar nombre de las columnas.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 168h
Riesgo en Desarrollo: N/A	Tiempo Real: 1 semana
Descripción: **1- Objetivo:** Editar el nombre de las columnas en la base de datos. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para editar el nombre de las columnas se debe tener en cuenta los siguientes datos: - Nombre de la columna y tipo de dato de la misma. **3- Flujo de la acción a realizar:** - El usuario selecciona la opción Edit Column e introduce los datos necesarios. - Si se selecciona el botón Cancel se cierra la ventana.	
Observaciones: El usuario debe tener una copia de respaldo y ser personal autorizado para acceder a la aplicación.	

Prototipo de interfaz:

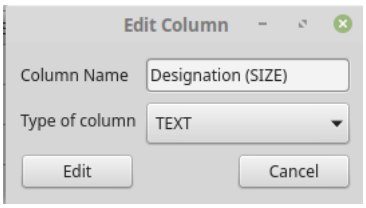


Tabla 2.5. Historia de Usuario RF №5.

Número: 5	Nombre del requisito: Eliminar tablas y/o columnas de la base de datos.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 168h
Riesgo en Desarrollo: N/A	Tiempo Real: 1 semana

Descripción:

1- Objetivo:

Eliminar tablas y/o columnas de la base de datos.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para eliminar el nombre de las tablas y/o columnas se debe tener en cuenta los siguientes datos:

- Nombre de la tabla o columna.

3- Flujo de la acción a realizar:

- El usuario selecciona la opción Remove Table o Remove Column.
- Si se selecciona el botón Cancel se cierra la ventana.

Observaciones: El usuario debe tener una copia de respaldo y ser personal autorizado para acceder a la aplicación.

Prototipo de interfaz:

Tabla 2.6. Historia de Usuario RF №6.

Número: 6	Nombre del requisito: Realizar la comunicación desde las interfaces de la aplicación.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas

Descripción:

1- Objetivo:

La base de datos debe comunicarse con las interfaces de la aplicación mediante un servidor local o remoto.

2- Acciones para lograr el objetivo (precondiciones y datos):
Para realizar la comunicación se debe tener en cuenta los siguientes datos:
- Host, Nombre de la base de datos, Nombre del usuario y Contraseña.

3- Flujo de la acción a realizar:
- El usuario introduce los datos para establecer la conexión y selecciona la opción Test Connection.
- Si se selecciona el botón Cancel se cierra la ventana.

Observaciones: Debe existir la conexión entre la base de datos y la aplicación.

Prototipo de interfaz:

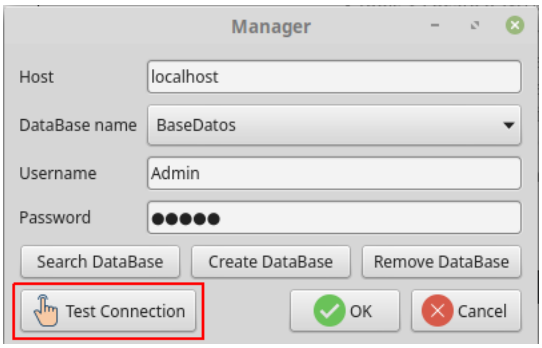


Tabla 2.7. Historia de Usuario RF №7.

Número: 7	Nombre del requisito: Implementar las funcionalidades para un servidor (local o remoto).
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas

Descripción:
1- Objetivo:
Se debe acceder a la base de datos desde la aplicación por un servidor remoto.

2- Acciones para lograr el objetivo (precondiciones y datos):
Para realizar la comunicación se debe tener en cuenta los siguientes datos:
- Host, Nombre de la base de datos, Nombre del usuario y Contraseña.

3- Flujo de la acción a realizar:
- El usuario introduce en el campo Host un ip proporcionado por la red.

Observaciones: Debe existir la conexión entre la base de datos y la aplicación.

Prototipo de interfaz:

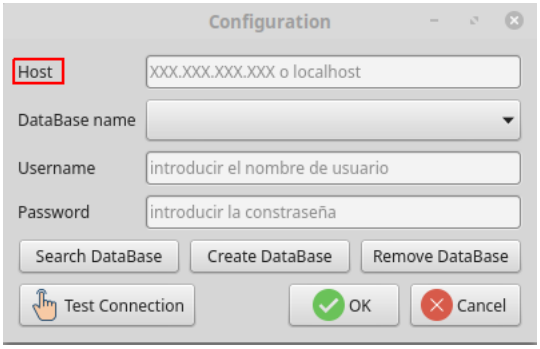


Tabla 2.8. Historia de Usuario RF №8.

Número: 8	Nombre del requisito: Insertar colecciones de componentes normalizados.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas

Descripción:

1- Objetivo:

Se debe insertar datos normalizados a la base de datos.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para realizar la acción se debe tener en cuenta los siguientes datos:

- Tornillos, tuercas, pasadores, arandelas y remaches.

3- Flujo de la acción a realizar:

- El usuario introduce los datos de las piezas normalizadas.

Observaciones: Debe estar creada la base de datos.

Prototipo de interfaz:

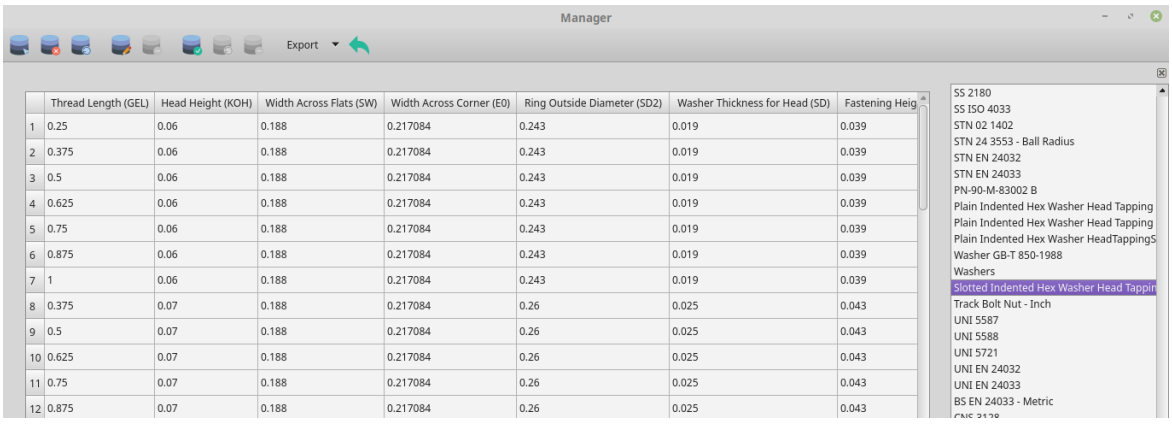


Tabla 2.9. Historia de Usuario RF №9.

Número: 9	Nombre del requisito: Insertar en las tablas columnas y filas.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas

Descripción:

1- Objetivo:

Las tablas deben estar compuestas por filas y columnas.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para realizar la acción se debe tener en cuenta los siguientes datos:

- En las columnas: características de la pieza y en la fila los datos.

3- Flujo de la acción a realizar:

- El usuario introduce los datos de las piezas normalizadas en filas y columnas.

Observaciones: Debe estar creada la base de datos.

Prototipo de interfaz:

Size Designation	File Name	Material	Part Number	Tipo
M3 x 8	AS 1427 - M3 x 8(5) III	Steel, Mild	AS 1427 - M3 x 8	Countersunk
M3 x 10	AS 1427 - M3 x 10(5) III	Steel, Mild	AS 1427 - M3 x 10	
M3 x 12	AS 1427 - M3 x 12(5) III	Steel, Mild	AS 1427 - M3 x 12	
M3.5 x 8	AS 1427 - M3.5 x 8(5) III	Steel, Mild	AS 1427 - M3.5 x 8	
M3.5 x 10	AS 1427 - M3.5 x 10(5) III	Steel, Mild	AS 1427 - M3.5 x 10	
M3.5 x 12	AS 1427 - M3.5 x 12(5) III	Steel, Mild	AS 1427 - M3.5 x 12	

Tabla 2.10. Historia de Usuario RF №10.

Número: 10	Nombre del requisito: Clasificar los datos de acuerdo a diferentes normas estándar.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 1
Prioridad: Alta	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas

Descripción:

1- Objetivo:

Los datos estarán clasificados de acuerdo a normas estándar.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para realizar la acción se debe tener en cuenta los siguientes datos:

- Clasificar en *(ANSI, BSI, GB, ISO, JIS, DIN, AS, IS, etc*.

3- Flujo de la acción a realizar:

- El usuario introduce los datos de las piezas ya normalizadas en filas y columnas.

Observaciones: Solo se realiza por el administrador.

Prototipo de interfaz:

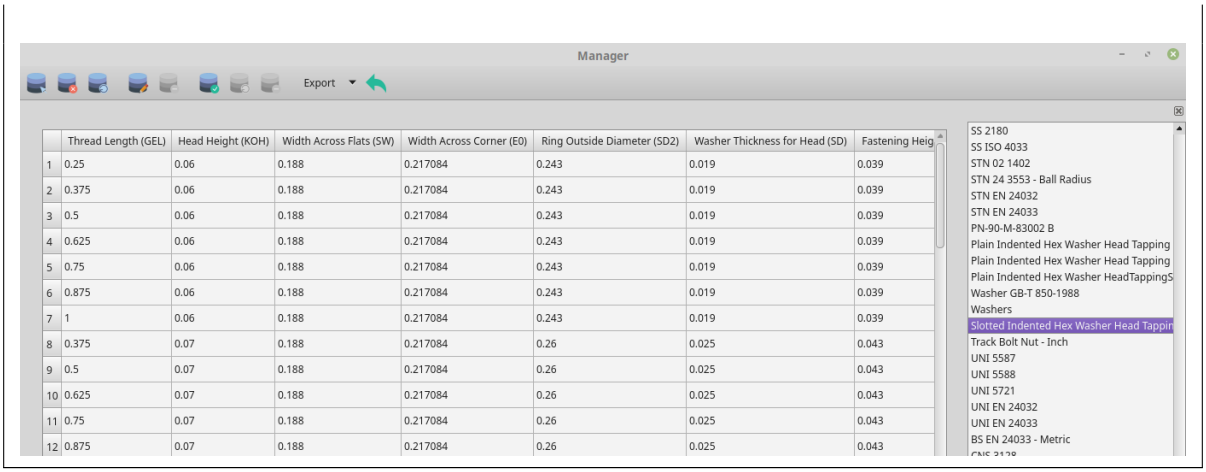


Tabla 2.11. Historia de Usuario RF №11.

Número: 11	Nombre del requisito: Exportar un contenido seleccionado en distintos formatos.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 2
Prioridad: Media	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas
Descripción: **1- Objetivo:** La base de datos debe tener funcionalidades para exportar en varios formatos. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para realizar la acción se debe tener en cuenta los siguientes datos: - Exportar un contenido seleccionado en formatos (xlsx, csv, ods, odf). **3- Flujo de la acción a realizar:** - El usuario selecciona una tabla y presiona el botón Export.	
Observaciones: Debe estar creada la base de datos.	
Prototipo de interfaz: 	

Tabla 2.12. Historia de Usuario RF №12.

Número: 12	Nombre del requisito: Crear la base de datos de acuerdo a roles definidos del sistema.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas

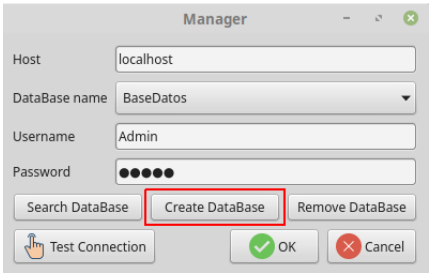
Descripción: **1- Objetivo:** La base de datos debe ser creada solo por el rol Admin. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para realizar la acción se debe tener en cuenta los siguientes datos: - Establecer la configuración de manera correcta con usuario Admin. **3- Flujo de la acción a realizar:** - El usuario selecciona la opción Create DataBase. - Si se selecciona el botón Cancel se cierra la ventana.
Observaciones: Debe estar establecida la conexión.
Prototipo de interfaz: 

Tabla 2.13. Historia de Usuario RF №13.

Número: 13	Nombre del requisito: Eliminar la base de datos de acuerdo a los roles del sistema.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 2
Prioridad: Alta	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas
Descripción: **1- Objetivo:** La base de datos debe ser eliminada solo por el rol Admin. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para realizar la acción se debe tener en cuenta los siguientes datos: - Establecer la configuración de manera correcta con usuario Admin. **3- Flujo de la acción a realizar:** - El usuario selecciona la opción Remove DataBase. - Si se selecciona el botón Cancel se cierra la ventana.	
Observaciones: Debe estar establecida la conexión.	
Prototipo de interfaz:	

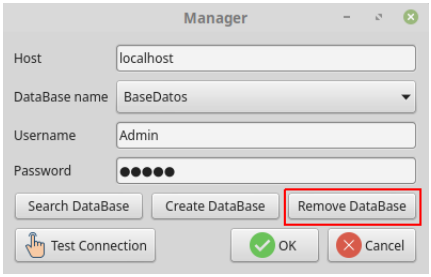


Tabla 2.14. Historia de Usuario RF №14.

Número: 14	Nombre del requisito: Realizar consultas, reportes y formularios.
Programador: Arlet Apezteguía Rigueira	Iteración Asignada: 2
Prioridad: Media	Tiempo Estimado: 336h
Riesgo en Desarrollo: N/A	Tiempo Real: 2, 5 semanas
Descripción: **1- Objetivo:** La base de datos debe garantizar consultas, reportes y formularios. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para realizar la acción se debe tener en cuenta los siguientes datos: - Los datos estén almacenados correctamente.	
Observaciones: Debe estar creada la base de datos.	

2.4. Análisis y Diseño

El diseño de software juega un papel importante en el desarrollo de software, permitiendo al ingeniero de software producir varios modelos del sistema que forman una especie de plan de la solución de la aplicación. La esencia del diseño del software es la toma de decisiones sobre la organización lógica del software. Algunas veces, se representa esta organización lógica como un modelo en un lenguaje definido de modelado tal como el Lenguaje Unificado de Modelado (UML) y otras veces simplemente se utiliza notaciones informales y esbozos para representar el diseño (Sommerville, 2005). El proceso de diseño tiene asociado la decisión del tipo arquitectura y los patrones de diseño que se emplea en el sistema, así como el diseño de diagramas que favorezcan el trabajo en la fase de implementación.

2.4.1. Estilo y patrón arquitectónico del software

Los estilos arquitectónicos definen componentes y las relaciones entre ellos, que pueden ser utilizados en instancias de este estilo, con un conjunto de restricciones en las descripciones arquitectónicas (ibíd.). El estilo elegido para el desarrollo del componente es Llamada y retorno ya que permite la modificación que se necesita en la mayoría de los sistemas y se reconoce por separar en la estructura del programa, usando una

jerarquía de control; donde un programa principal puede llamar a varios componentes del programa, que a su vez pueden llamar a otros componentes. La arquitectura de software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (Sommerville, 2005). El patrón arquitectónico elegido es *Model-View-Controller* (Modelo-Vista-Controlador). El patrón Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en el usuario en tres clases diferentes (ver Figura 2.3) (ibíd.).

- El modelo representa los datos del programa, no tiene conocimiento de los controladores o de las vistas, es el propio sistema el que tiene la responsabilidad de mantener relaciones entre el modelo y sus vistas, y notificar a las vistas cuando exista un cambio en el modelo.
- La vista maneja la visualización de la información representados en el modelo.
- El controlador es el resultado de las acciones del usuario, informando al modelo y/o a la vista si se solicita un cambio.

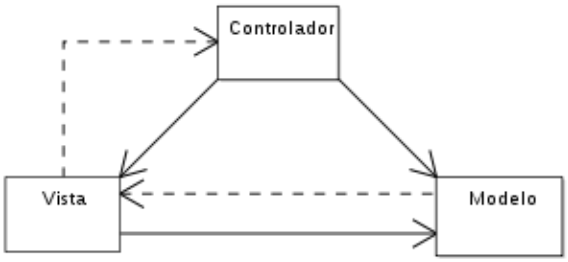


Figura 2.3. Arquitectura Modelo-Vista-Controlador.

2.4.2. Diagrama de clases del diseño

Los diagramas de clase en Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) pueden usarse cuando se desarrolla un modelo de sistema para mostrar las clases en un sistema y las relaciones entre dichas clases (Larman, 2004) (ver Figura 2.4). El componente cuenta con 5 clases y 16 funcionalidades:

- *Configuration* (encargada de establecer la conexión con la base de datos).
- *DataBaseManger* (encargada de contener las consultas a los datos y brindar la conexión).
- *Manager* (interfaz visual).
- *AddColumn* (encargada de adicionar columnas en las tablas de la base de datos).
- *EditColumn* (encargada de editar columnas en las tablas de la base de datos).

2.4.3. Patrones de diseño del software

Provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos (Sommerville, 2005). Los patrones son una descripción

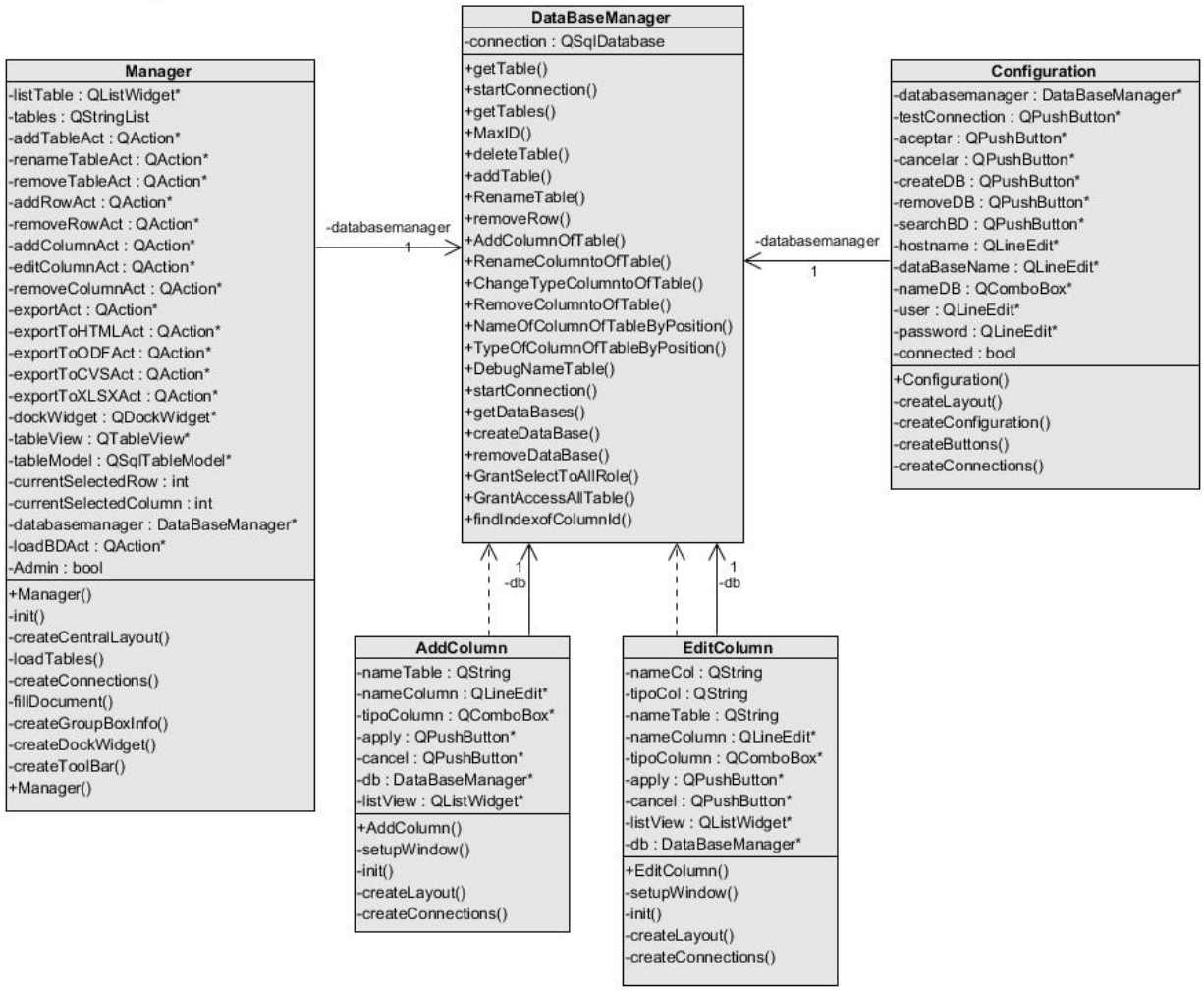


Figura 2.4. Diagrama de clases.

de un problema y la solución, con un nombre que codifican buenos consejos y principios relacionados con frecuencia con la asignación de responsabilidades (Larman, 2004).

Los patrones GRASP *(General Responsibility Assignment Software Patterns)* en español patrones generales de software para asignar responsabilidades, estos patrones GRASP fueron elegidos para diseñar con éxito el software:

- Experto en Información.
- Creador.
- Bajo Acoplamiento.
- Alta Cohesión.
- Controlador.

Experto: Se encarga de asignar una responsabilidad al experto en información, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Es la asignación de responsabilidades a cada clase, facilitando que los sistemas sean más fáciles de entender, mantener y ampliar. Este patrón se evidencia en la clase *databasemanager*, que se encarga de tener la información referente a los componentes normalizados.

Creador: Se encarga de asignar a las clases la responsabilidad de crear una instancia de otra. Soportando mayor claridad y encapsulamiento. La intención del patrón Creador es encontrar un creador que necesite conectarse al objeto creado. Este patrón se evidencia en la clase *databasemanager*, pues la misma es la encargada de crear instancias.

Bajo Acoplamiento: Se encarga de asignar una responsabilidad para mantener el bajo acoplamiento. Mantener las clases lo menos relacionadas posibles, es la medida de la fuerza con que una clase está conectada a otras. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios. Este patrón se evidencia en las clases *addcolumn y configuration* ya que no existe relación entre ellas.

Alta Cohesión: Se encarga de asignar una responsabilidad de modo que la cohesión siga siendo alta. Es la forma de medir cuán relacionadas están las responsabilidades de una clase. Una clase con alta cohesión es fácil de entender, reutilizar y mantener. Este patrón se evidencia en la clase *configuration* ya que pueden trabajar con la información de la base de datos sin una gran cantidad de trabajo.

Controlador: Se encarga de asignar la responsabilidad del manejo de los eventos de un sistema a una clase. Es un objeto de interfaz no destinado al usuario que se encarga de manejar un evento del sistema. Este patrón se evidencia en la clase *manager*, que es la encargada de atender los eventos del sistema que son generados por un actor externo.

Patrón GOF (pandilla de los cuatro o *Gang-of-Four*): El patrón que se evidencia es el Observador, ya que define una dependencia entre objetos, de forma tal que cuando el objeto cambia de estado, todos sus dependientes son notificados. Se usará el patrón observador cuando un elemento quiere estar pendiente de otro, sin tener que estar comprobando de forma continua si ha cambiado o no.

2.4.4. Diseño de la base de datos

El diseño de una base de datos consiste en definir la estructura de los datos de un sistema. Para un correcto diseño es necesario abordar cada una de las fases en el proceso de diseño, definiendo para ello el modelo conceptual, el lógico y el físico. Una base de datos diseñada de forma correcta le proporciona acceso a información actualizada y precisa. La imagen muestra un fragmento del diseño, la base de datos cuenta con un total de 300 tablas, la entidad *Fasteners* (elementos de fijación) es la principal, esta abarca las diferentes variedades de piezas normalizadas como *bolts* (tornillos), *nuts* (tuercas), *pins* (pasadores), *rivets* (remaches) y *washers* (arandelas). Estas son las entidades padres de la cual heredan 100 bolts, 110 nuts, 17 rivets, 21 washers y 52 pins (ver Figura 2.5) (Lozada, s.f.).

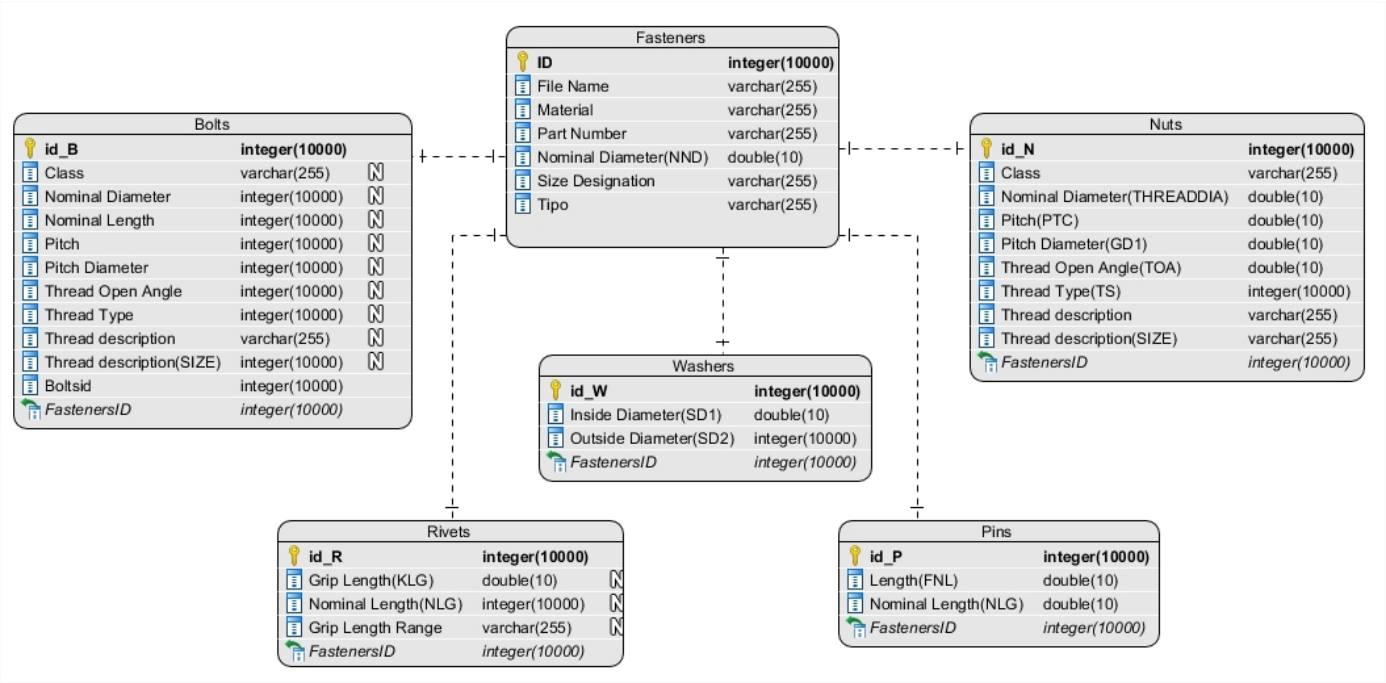


Figura 2.5. Modelo Entidad-Relación.

2.5. Conclusiones del capítulo

- El resultado de la propuesta solución se ajusta al problema de investigación, dando cumplimiento a los requisitos funcionales y no funcionales del software.
- El diagrama de clases permitió perfeccionar el diseño de la estructura del componente, para su óptimo funcionamiento.
- Del análisis realizado del diagrama entidad-relación se lograron identificar las relaciones de la propuesta de solución, proponiendo una visión clara de la representación.

CAPÍTULO 3

Implementación y pruebas

En este capítulo se realiza una valoración de los resultados obtenidos durante el proceso de implementación y pruebas. Se abordan aspectos importantes asociados a los estándares de codificación, garantizando buenas prácticas en la lectura del código generado. Además se hace alusión a las pruebas realizadas a la propuesta de solución para evitar fallos posteriores.

3.1. Implementación

La fase de implementación del software posee como entradas los artefactos de la fase anterior (diseño), como: diagramas de clases, especificación de arquitectura, patrones a emplear en el sistema. En la implementación se define el estándar de codificación a emplear y se realizan las implementaciones a las historias de usuarios. En la implementación del componente se emplea una *Application Programming Interface* (API) con el nombre *Manager*. Se utilizó con el fin de que el usuario pueda acceder a la base de datos desde la aplicación, sin tener que utilizar otro programa. El componente cuenta con un total de 5 ficheros fuente (.cpp) y 5 ficheros cabecera (.h), con un total de 16 funcionalidades, 1442 líneas de código y un total de 259 líneas en blanco.

3.1.1. Estándar de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo regular, como si se hubiese realizado por un programador. La forma usada va a depender de la facilidad para entender el código y retomar ciertas partes realizadas por otros integrantes, así como la eliminación de las mismas. La elaboración y utilización de estos estándares permiten a todos los desarrolladores realizar la implementación siguiendo las pautas predefinidas y así el resto de los desarrolladores puedan entender el código, facilitando que un proyecto de software se convierta en un sistema legible, uniforme y fácil de mantener. (ver Figuras 3.1, 3.2, 3.4) (ASPL, 2018).

Descripción	Ejemplo
Definición de Objetos, Clases, funciones y atributos	

Todos los nombres de las clases implementadas comenzarán con letra mayúscula. En caso de poseer un nombre compuesto se escribirán de acuerdo a la normativa CamelCase-UpperCamelCase.	class Foo{
	cuerpo de la clase
	}
	class FooFirst{
	cuerpo de la clase
	}

Siempre se declara para todas las clases implementadas su respectivo destructor de clase.	virtual ~Foo()
La declaración de funciones o métodos siempre comenzarán en letra inicial minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato retorno> funcion() <Tipo dato retorno> funcionCompuesta() <Tipo dato retorno>funcionDobleCompuesta()

Figura 3.1. Estándar de codificación.

Los atributos siempre estarán escritos con letra minúscula. En caso de ser un nombre compuesto se regirá por la normativa CamelCase-lowerCamelCase.	<Tipo dato> atributo; <Tipo dato> atributoNombreCompuesto;
Definición de parámetros dentro de las funciones y constructores de clases	
Los nombres de los identificadores de los parámetros en las funciones deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCaselowerCamelCase.	<Tipo dato retorno> funcion(<tipo><id1>, <tipo><id2>, <tipo><idN>)
Los identificadores de los parámetros dentro de los constructores de las clases deben estar escritos con minúscula separados a un espacio cada uno y en caso de ser compuesto utilizar normativa CamelCase-lowerCamelCase.	Clase(<tipo><id1>, <tipo><id2>, <tipo><idN>)
Definición de expresiones	

Para una mejor comprensión en la lectura y legibilidad del código los operadores binarios exceptuando los punteros, función de llamado a miembros, escritura de un arreglo y paréntesis de una función se escribirán con un espacio entre ellos	x + y;
	x == y;
	idFuncion.miembro();
	idFuncion->miembro();
	array[];

Figura 3.2. Estándar de codificación.

Ejemplo de la normativa *CamelCase* en el código de la clase *Configuration*.

Definición de estructuras de control y bucles	

Las estructuras de control y los bucles estarán definidos de igual manera en ambos casos siguiendo el estándar determinado por el *framework* de Qt.	Para las estructuras if, else, if else :
	<estructura control>(condición){
	tarea a ejecutar
	}
	Para los bucles while, for, do while y otros:
	<bucle>(condiciones){
	tarea a ejecutar
	}

Comentarios en el código según el estándar de C++.	
Comentarios pequeños.	/* comentario sencillo */
Otros comentarios.	/* *Comentario */
Comentario de versión, descripción de clase y otras características de la clase o paquete.	/* ********** *Comentario amplio * ********** */

Figura 3.3. Estándar de codificación.

```
void Configuration::isConnected()
{
    QString  Hostname  = hostname->text();
    QString dataBase = nameDB->currentText();
    QString User = user->text();
    QString Password =  password->text();
    if(User == "Invitado")
        Password = "123";

    if(Hostname.isEmpty() || dataBase.isEmpty() || User.isEmpty() || Password.isEmpty())
    {  QMessageBox::information(this,"Error","Empty fields") ;
        return;}
    connected = databasemanager->startConnection(Hostname,dataBase,User,Password);
    if(connected)
        QMessageBox::information(this,"Connection","There is connection to the Data Base");
    else
        QMessageBox::information(this,"Connection","There is no connection with the Data Base");
}
```

Figura 3.4

3.1.2. Diagrama de despliegue

Un diagrama de despliegue es la forma de mostrar cómo los componentes de software se despliegan físicamente en los procesadores; es decir, el diagrama de despliegue muestra el hardware y el software en el sistema (Sommerville, 2011).

El diagrama de despliegue que se muestra a continuación representa la distribución física del sistema a través de nodos. Está compuesto por una PC Cliente que deberá tener instalado la aplicación Ingeniero, donde la comunicación entre ella y el servidor de base datos PostgreSQL se lleva a cabo mediante el Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP) utilizando el puerto 5432 (ver Figura 3.5).

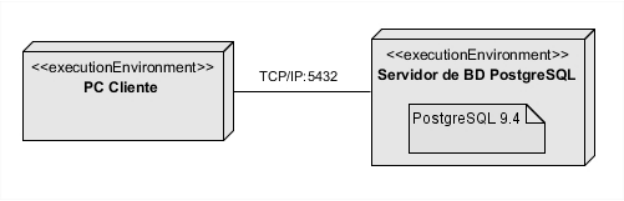


Figura 3.5. Diagrama de despliegue.

3.1.3. Resultados de la implementación

En este apartado se muestran algunas imágenes sobre el resultado de la implementación. En las imágenes 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12 se hace referencia a la *Application Programming Interface*, mediante la utilización de la interfaz el usuario puede agregar, editar o eliminar tabla, fila o columna, así como exportar en diferentes formatos.

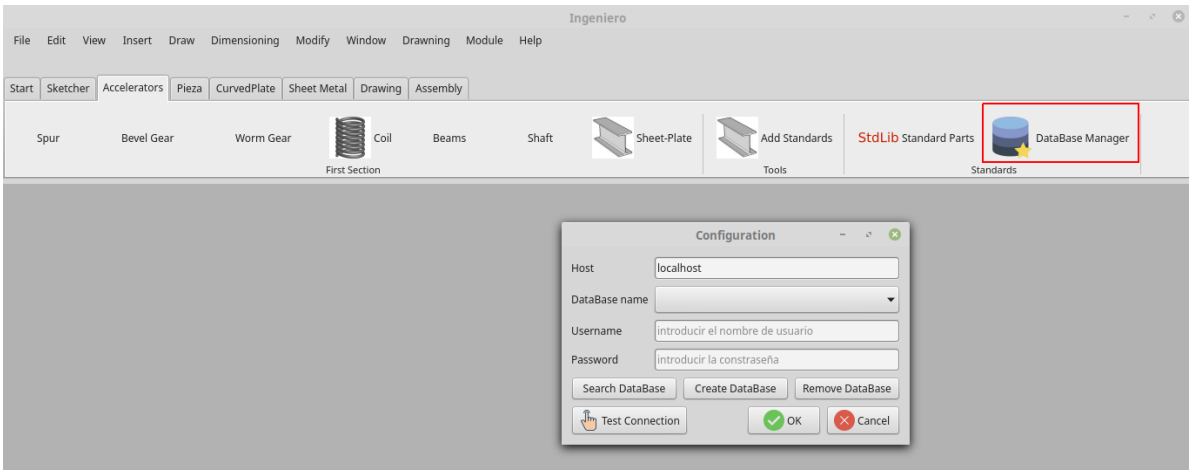


Figura 3.6. Funcionalidad para realizar la configuración.

Figura 3.7. Interfaz de la aplicación.

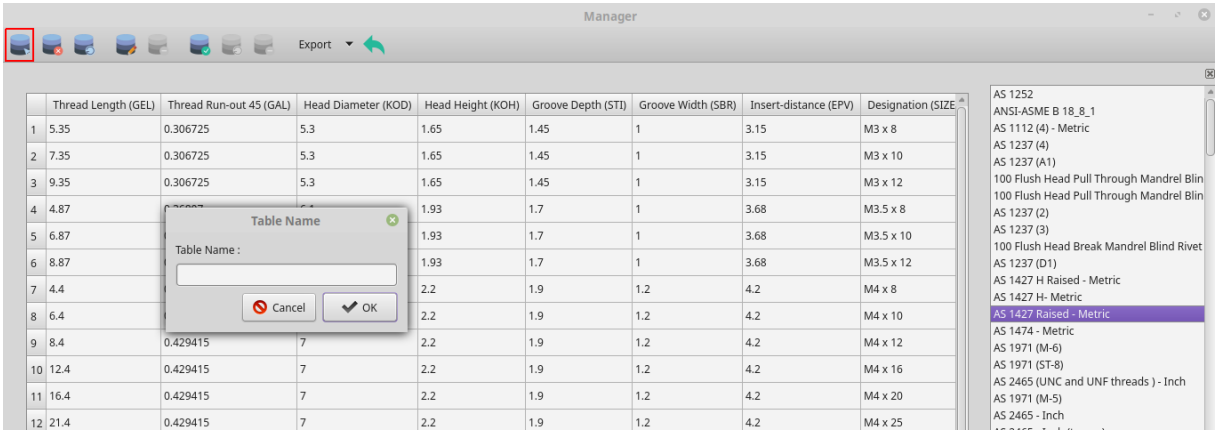


Figura 3.8. Funcionalidad adicionar tabla.

Figura 3.9. Funcionalidad editar nombre de la tabla.

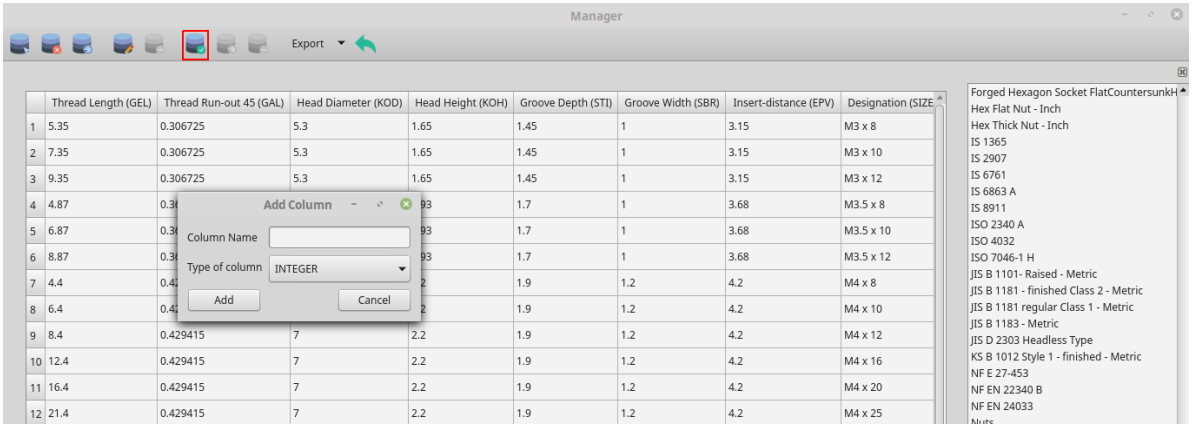


Figura 3.10. Funcionalidad adicionar columna.

Figura 3.11. Funcionalidad exportar.

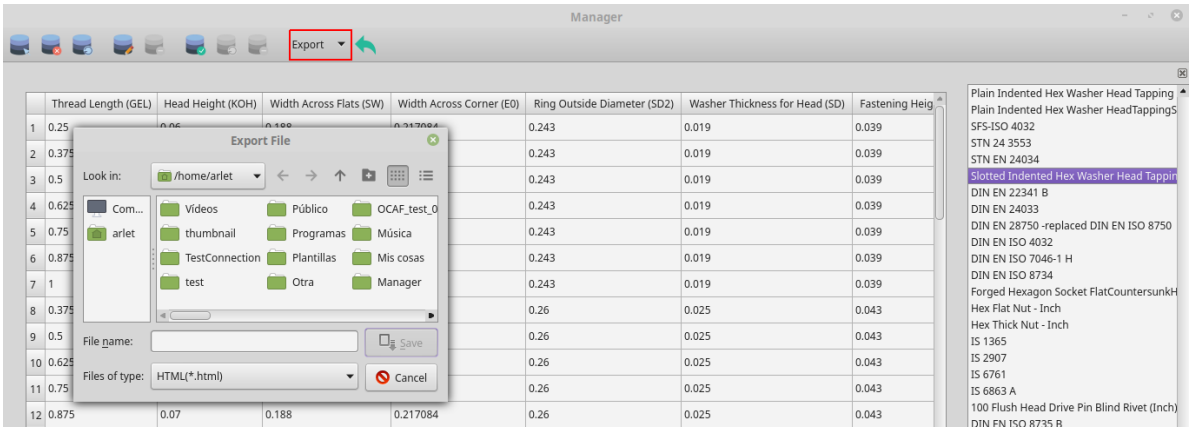


Figura 3.12. Vista para guardar el archivo exportado.

3.2. Pruebas de software

Las pruebas constituyen “Una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados y se realiza una evaluación de algún aspecto del sistema o componente (Pressman, 2005).”

Según plantea (ibíd.) una vez que se ha generado el código comienzan las pruebas del programa. El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales; es decir, realizar las pruebas para la detección de errores y asegurar que la entrada definida produce resultados reales de acuerdo con los resultados requeridos.

Las pruebas de software según (Sommerville, 2011) permiten verificar si el software funciona según lo esperado, cumpliendo con dos objetivos fundamentales:

- Demostrar al desarrollador y al cliente que el software satisface sus expectativas a partir de los requisitos planteados.
- Determinar las imperfecciones del software que impide que funcione correctamente.

Las pruebas intentan demostrar que un programa funciona según lo esperado, además descubrir defectos en el programa antes de ser utilizado, las pruebas son realizadas de acuerdo a planes de prueba o modelo V como son (Sommerville, 2011). A continuación se describen las pruebas realizadas al componente de datos para la gestión de piezas normalizadas de la aplicación CAD "Ingeniero"

- Las pruebas de unidad se enfocan en comprobar la funcionalidad de objetos o métodos. Se emplea para detectar errores debidos a operaciones incorrectas realizadas.
- Las pruebas de integración valoran si los componentes individuales trabajan en conjunto tal y como se espera. Se prueba el funcionamiento de los diferentes módulos del sistema una vez unidos, verificando el comportamiento de los mismos frente a las comunicaciones que se produzcan entre ellos. El objetivo es detectar errores y comprobar el correcto funcionamiento de los componentes como un todo.
- Las pruebas funcionales está constituida por una serie de pruebas diferentes cuyo objetivo es validar que el software cumple con todos los requisitos especificados por el cliente.
- Las pruebas de aceptación: es la realización de una serie de pruebas que demuestran la satisfacción del cliente para aceptar o no el sistema.

3.2.1. Métodos de prueba

Las pruebas funcionales según (ibíd.) se realizan con el objetivo de comprobar que el software o sistema funcione de forma correcta. Para lograr este objetivo es necesario conocer los siguientes métodos de prueba:

- **Pruebas de caja negra:** son aquellas pruebas que se realizan sobre la interfaz del software. El objetivo es evaluar que las funciones del software funcionen correctamente, a partir del uso de los casos de prueba (no se ve el código). Para desarrollar las pruebas de caja negra existen varias técnicas (Técnica de partición de equivalencia, Técnica de análisis y valor límite, Técnica de Grafos de causa de efecto).

- **Pruebas de caja blanca:** son aquellas pruebas que comprueban los caminos lógicos del software. Se realiza sobre el código para determinar si el estado real del programa coincide con el esperado.

Con el objetivo de aplicar las pruebas de caja negra, es necesario apoyarse en el Diseño de casos de prueba de la técnica de partición de equivalencia propuesto por la metodología de desarrollo de software seleccionada. Un caso de prueba es una forma de comprobar el correcto funcionamiento del sistema, en estos se incluyen las entradas, respuesta del sistema y condiciones con la que se ha de verificar, constituyendo la guía principal para el probador.

3.2.2. Diseño de Casos de Prueba

Los casos de prueba son un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y postcondiciones de ejecución, para verificar el cumplimiento de un requisito determinado (ibíd.).

Son una herramienta manual y se diseñan a partir de los Casos de uso, Historias de usuarios o Requisitos para verificar que la aplicación es satisfactoria y encontrar las deficiencias del software. A continuación se muestran ejemplos de casos de prueba correspondiente los requisitos (ver Tablas 3.1, 3.2 y Anexos A.1).

Tabla 3.1. Diseño de Casos de Prueba RF №1

Descripción general				
El sistema debe permitir adicionar tablas a la base de datos.				
SC 1 Adicionar tablas.				
Escenario	Descripción	TableName	Respuesta del sistema	Flujo central
EC 1.1 Adicionar tablas.	Selecciona la pestaña Add Table la cual muestra una sección para adicionar la tabla a la base de datos.	Bolts.	Se adiciona una tabla con los campos especificados a la base de datos.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager/ Add Table/botón aceptar.
EC 1.2 Adicionar tablas presionando el botón cancelar.	Se cancela la operación para adicionar una tabla.	Bolts.	No se adiciona la tabla.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager Add Table/botón cancelar.
EC 1.3 Adicionar tabla con campo Table Name vacío.	Se intenta adicionar una tabla dejando campos vacíos.	(vacío)	Detecta que el campo Table Name está vacío, muestra un mensaje de error y no se adiciona la tabla.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager/ Add Table/botón aceptar.

Tabla 3.2. Diseño de Casos de Prueba RF №2

Descripción general					
El sistema debe permitir adicionar columnas a la base de datos.					
SC 1 Adicionar columnas.					
Escenario	Descripción	ColumnName	Typeofcolumn	Respuesta del sistema	Flujo central

EC 1.1 Adicionar una columna.	Selecciona la pestaña Add Column la cual muestra una sección para adicionar columna a la base de datos.	Designation.	Integer.	Se adiciona una columna con los campos especificados a la base de datos.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Add Column/ botón adicionar.
EC 1.2 Adicionar una columna presionando el botón cancelar.	Se cancela la operación para adicionar una columna.	Designation.	Integer.	No se adiciona la columna.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Add Column/ botón cancelar.
EC 1.3 Adicionar columna con campo Column Name vacío.	Se intenta adicionar una columna dejando campos vacíos.	(vacío).	Integer.	Detecta que el campo Column Name está vacío, muestra un mensaje de error y no se adiciona la columna.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Add Column/ botón adicionar.

3.2.3. Resultados obtenidos

Resultados de las pruebas funcionales

Al componente de gestión de datos de piezas normalizadas se le realizaron pruebas funcionales para identificar los posibles fallos y no conformidades presentes en la propuesta de solución. Se realizaron tres iteraciones mediante las pruebas de caja negra las cuales identificaron las siguientes no conformidades (ver Tabla 3.3 y Figura 3.14).

Tabla 3.3. No Conformidades del sistema

No. NC	Requisito Funcional	Descripción	Complejidad	Estado
1	RF 1	No insertaba más de una fila.	Baja	Resuelta
2	RF 1	No mostraba mensaje de error cuando habían campos vacíos.	Baja	Resuelta
3	RF 1	Se eliminaba el valor de la fila anterior al insertar una fila nueva.	Baja	Resuelta
4	RF 2	No se adicionaba la columna a la tabla seleccionada.	Media	Resuelta
5	RF 3	No se modificaba el nombre de la tabla seleccionada.	Baja	Resuelta

6	RF 6	No se conectaba al servidor remoto.	Alta	Resuelta
7	RF 10	No exportaba en formato HTML.	Media	Resuelta
8	RF 11	Cualquier usuario tenía acceso a modificar la base de datos.	Alta	Resuelta
9	RF 12	Cualquier usuario tenía acceso a eliminar la base de datos.	Alta	Resuelta

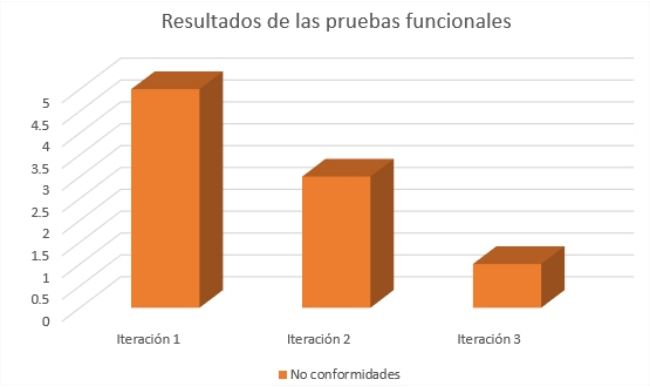


Figura 3.13. Cantidad de No Conformidades por iteración.

Resultado de las pruebas unitarias

Para la realización de las pruebas unitarias se empleó Qt Test, el cual provee clases y bibliotecas de QT para realizar dichas pruebas. Serán ejecutados por el marco de prueba y pueden usarse para inicializar y limpiar la prueba completa o la función de prueba actual. Se usa *initTestCase* antes de que se ejecute la primera función de prueba y *cleanupTestCase* después de que se ejecutó la última función de prueba (*Qt Test* 2018).

Código fuente 3.1. Código para ejecutar pruebas unitarias

```
#include <QObject>
#include <QTest>
#include <databasemanager.h>
class TestDataBaseManager : public QObject
{
    Q_OBJECT
  private slots:
    void TestConnection();
    void TestGetTable();
    void TestMaxID();

private:
    DataBaseManager * db;
```

```
};

void TestDataBaseManager::TestConnection()
{
      db = new DataBaseManager();
      bool result = db->startConnection("localhost","BaseDatos","
         postgres","1111");
      QCOMPARE(result , true);
}

void TestDataBaseManager::TestGetTable()
{
     //bool result = db->startConnection("localhost","BaseDatos","
        postgres","1111");
     QVERIFY(!db->getTables().isEmpty());
}

void TestDataBaseManager::TestMaxID()
{
       QCOMPARE( db->MaxID("100 Flush Head Break Mandrel Blind Rivet (
          Inch) (IFI)") , 67209 );
}
```

```
********* Start testing of TestDataBaseManager *********
Config: Using QtTest library 5.9.5, Qt 5.9.5 (x86_64-little_endian-lp64 shared (dynamic) release build; by GCC 7.3.0)
PASS   : TestDataBaseManager::initTestCase()
PASS   : TestDataBaseManager::TestConnection()
PASS   : TestDataBaseManager::TestGetTable()
PASS   : TestDataBaseManager::TestMaxID()
PASS   : TestDataBaseManager::cleanupTestCase()
Totals: 5 passed, 0 failed, 0 skipped, 0 blacklisted, 34ms
********* Finished testing of TestDataBaseManager *********
```

Figura 3.14. Resultado de las pruebas unitarias.

Resultado de las pruebas de integración

Estas pruebas valoran si los componentes individuales trabajan en conjunto tal y como se espera de ellos. En las pruebas de integración se utilizó la estrategia Big-Bang, ya que es la única estrategia de pruebas integradas que no es incremental. La estrategia Big-Bang se integra únicamente en el momento en el que se dispone de todos los componentes (Pressman, 2005).

3.2.4. Pruebas de Aceptación

Las pruebas de aceptación se realizan cuando el cliente prueba un sistema para decidir si cumple con todos los requisitos establecidos por el cliente que lo desea, o si se requiere más desarrollo (Sommerville, 2011). Se realizó una entrevista al cliente y a los estudiantes del grupo de investigación para comprobar la aceptación del componente, con esta entrevista el componente obtuvo un resultado de 98% de satisfacción entre el personal, debido a que la forma de verificar que funciona completamente bien se realiza por el módulo “Maqueta con funcionalidades para acelerar el diseño de tornillos empleando los contenidos de una Base de datos”, realizado por otro autor.

3.3. Conclusiones del capítulo

- Los estándares de codificación permiten una mejor comprensión y limpieza del código utilizado para la implementación.
- Se identificaron un total de 9 no conformidades, las cuales han sido resueltas garantizando que el cliente obtenga una solución libre de fallos.
- Los resultados de las pruebas evidencian que el componente con un 98% de satisfacción, se encuentra listo para ser utilizado en la aplicación CAD “Ingeniero”.

Conclusiones

Con la realización de este trabajo se arribaron a las siguientes conclusiones:

- Los fundamentos teóricos-metodológicos permitieron inferir que los sistemas comerciales de diseño según la tendencia emplea lenguaje SQL, lo que resulta factible utilizar una base de datos relacional para una mejor organización de la información.
- La etapa de análisis y diseño guiada por la metodología AUP-UCI permitió obtener un conjunto de artefactos que justifican el proceso de desarrollo y la documentación técnica asociada a la propuesta de solución.
- El resultado de la implementación incidió positivamente en la aplicación de diseño asistido por computadora “Ingeniero”, aportando un componente que acelera el diseño de piezas normalizadas.
- Las técnicas de validación aplicadas, evidencian que la solución con un 98% de satisfacción, se encuentra lista para ser utilizada en la práctica de la ingeniería en el país.

Recomendaciones

A partir de los resultados obtenidos en la presente investigación se recomienda al grupo de investigación:

- Continuar haciendo pruebas para una validación más completa del componente.
- Extender la librería de componentes estándar a otras ramas de la ingeniería para elementos estructurales como vigas y piezas de construcción naval.

Referencias bibliográficas

ACENS, 2017. Bases de datos NoSQL. En: *Bases de datos NoSQL*. Url: https://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf (vid. pág. 9).

ALONSO, 2012. *2 Bases de datos relacionales*. Url: https://www.um.es/geograf/sigmur/temariohtml/node63_mn.html (vid. pág. 6).

ÁLVAREZ, Orlando Farray, 2018. *Elementos de la Metodología de Investigación para la realización del diseño teórico-metodológico de los trabajos de diploma en la UCI*. científico. Universidad de las Ciencias Informáticas (vid. pág. 4).

ASPL, 2018. *Etándares de codificación*. Url: url:%20http%20:%20/%20/%20www%20.%20aspl%20.%20es%20/%20fact%20/%20files%20/%20aspl%20-%20fact%20/%20estandares-node2.html (vid. pág. 42).

Autodesk Inventor Professional. Url: https://www.imaginit.com/software/autodesk-products/inventor%20and%20https://www.3ds.com/es/productos-y-servicios/catia/ (vid. págs. 2, 21).

Beneficios de la normalización. Url: http://www.unit.org.uy/normalizacion/beneficios/ (vid. pág. 20).

CARRASCO, José Lázaro Román, 2018. Atandelas. Url: https://comunidad.leroymerlin.es/t5/Bricopedia-Bricolaje/Qu%C3%A9-tipo-de-arandelas-existen-y-para-qu%C3%A9-se-usan/ta-p/89977 (vid. pág. 18).

CARVAJAL, Alejandro Manuel Rubinos y LEÓN, Heidy Alina Nuevo, 2011. Seguridad en bases de datos. *Revista Cubana de Ciencias Informáticas*. Vol. 5, n.º 1, págs. 17. ISSN 1994-1536. Url: https://www.redalyc.org/articulo.oa?id=378343671005 (vid. pág. 3).

CISNEROS, Eduardo, 2017. Estandarización de sistemas. En: *Estandarización de sistemas*. Url: https://innovando.net/estandarizacion-de-componentes-de-sistemas-automatizados-1/ (vid. pág. 15).

DÍAZ, Alejandro Gutiérrez, 2014. *Bases de datos*. Url: https://www.aiu.edu/cursos/base%20de%20datos/pdf%20leccion%201/lecci%C3%B3n%201.pdf (vid. págs. 2, 7-9, 11).

Funciones de gestión de datos integradas en Solid Edge, 2017. Url: https://www.pixelsistemas.com/solid-edge (vid. pág. 21).

GRADO, Carlos García, 2015. *Solidworks para diseño y dibujo mecánico* (vid. pág. 20).

Información general de modelado de datos, 2010. Url: https://sites.google.com/site/jalexiscv/modelosdedatos (vid. pág. 13).

Introducción a los Sistemas CAD/CAM/CAE. Url: http://ocw.uv.es/ingenieria-y-arquitectura/expresion-grafica/eg_tema_2.pdf (vid. págs. 2, 6, 7).

KERN, Vinicius Medina, 1994. *Database Systems for CAD*. Url: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.8744&rep=rep1&type=pdf. Department of Industrial y Systems Engineering (vid. págs. 1, 2).

LARMAN, Craig, 2004. *UML y Patrones*. Segunda edición (vid. págs. 38, 39).

LÓPEZ, Ricardo Gutiérrez, 2017. *Mecánica técnica* (vid. pág. 17).

LOZADA, Alexys. *Las etapas del diseño de una correcta base de datos relacional*. Url: https://ed.team/blog/las-etapas-del-diseno-de-una-correcta-base-de-datos-relacional (vid. pág. 40).

MELGAREJO, Álvaro Irles y MOYA, Álvaro Outerio. *Sistemas Gestores de Bases de Datos*. Url: http://gplsi.dlsi.ua.es/bbdd/bd1/lib/exe/fetch.php?media=bd1:0910:trabajos:aimsgbd.pdf (vid. pág. 12).

MESA, Miriam, 2011. *Base de datos*. Url: https://miriammeza.wordpress.com/2011/02/17/ventajas-y-desventajas-de-las-bdoo/ (vid. pág. 9).

MICHEL, Erodis Pérez, 2015. *Consideraciones para implementar una Base de datos de modelos CAD*. Url: https://www.researchgate.net/publication/309592256_CONSIDERACIONES_PARA_IMPLEMENTAR_UNA_BASE_DE_DATOS_DE_MODELOS_CAD (vid. pág. 1).

OCA MONTANO, José Luis Montes de, 2015. La migración hacia software libre en Cuba: complejo conjunto de factores sociales y tecnológicos en el camino de la soberanía nacional. *Revista Universidad y Sociedad*. Vol. 7, págs. 120-125. Url: http://rus.ucf.edu.cu%20and%20https://www.researchgate.net/publication/305317510_La_Migracion_hacia_software_libre_en_Cuba_Complejo_conjunto_de_factores_sociales_y_tecnologicos_en_el_camino_de_la_soberania_nacional (vid. pág. 2).

PARÉ, Rafael Camps; SANTILLÁN, Luis Alberto Casillas; COSTA, Dolors Costal; GINESTÁ, Marc Gibert y CARME MARTÍN ESCOFET, Oscar Pérez Mora, *Bases de datos*. Url: https://www.uoc.edu/masters/oficiales/img/913.pdf (vid. pág. 13).

Pasos para el proceso de diseño de una base de datos. Url: http://informaticajonathan.blogspot.com/2010/04/pasos-para-el-proceso-de-diseno-de-una.html (vid. pág. 14).

PENADÉS, Patricio Letelier, 2013. *Métodologías ágiles para el desarrollo de software:Extreme Programming* (vid. pág. 28).

PRESSMAN, Roger S., 2005. *Ingeniería del Software.Un enfoque práctico*. Quinta edición and Séptima edición (vid. págs. 47, 52).

Qt Test, 2018. Url: https://doc.qt.io/qt-5/qttest-index.html (vid. pág. 51).

SÁNCHEZ, Tamara Rodríguez, 2015. *Metodología de desarrollo para la actividad productiva de la UCI*. Universidad de las Ciencias Informáticas (vid. pág. 22).

SILBERSCHATZ, Abraham; KORTH, Henry F. y SUDARSHAN, S., 2002. *Fundamentos de Bases de Datos*. Url: http://artemisa.unicauca.edu.co/~cardila/Libro_Silberschatz.pdf (vid. págs. 13, 14).

Sitio Oficial ISO. Url: http://www.iso.org/iso/home.html (vid. págs. 2, 19, 20).

Solid Edge, 2015. Url: http://majentaplm.com/wp-content/uploads/ST8-Solid-Edge-Readme-File.pdf (vid. pág. 21).

SOMMERVILLE, Ian, 2005. *Ingeniería del software: Requerimientos*. Séptima edición. ISBN 84-7829-074-5 (vid. págs. 26, 27, 37, 38).

SOMMERVILLE, Ian, 2011. *Ingeniería del Software*. Novena edición. ISBN 978-607-32-0603-7 (vid. págs. 44, 47, 48, 52).

Tipos de remaches para uniones perfectas, url: https://sumatec.co/tipos-de-remaches-para-uniones-perfectas/ (vid. pág. 18).

Tuercas y Tornillos, 2018 (vid. pág. 16).

Generado con LaTeX: 7 de junio de 2019: 9:41am

Apéndices

APÉNDICE A

Anexos

A.1. Casos de prueba

Tabla A.1. Diseño de Casos de Prueba RF №3

Descripción general				
El sistema debe permitir editar los nombres de las tablas				
SC 1 Editar tablas.				
Escenario	Descripción	TableName	Respuesta del sistema	Flujo central
EC 1.1 Editar tablas.	Selecciona la pestaña Rename Table la cual muestra una sección para editar la tabla en la base de datos.	DIN 917	Se edita una tabla con los campos especificados.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/Manager/ Rename Table/botón aceptar.
EC 1.2 Editar tablas presionando el botón cancelar.	Se cancela la opción para editar una tabla.	DIN 917.	No se edita la tabla con los campos especificados	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/Manager/ Rename Table/botón cancelar.
EC 1.3 Editar tabla con campo Table Name vacío.	Selecciona la opción dejando campos vacíos.	(vacío).	Detecta que el campo Table Name está vacío y no se edita la tabla.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/Manager/ Rename Table/botón aceptar.

Tabla A.2. Diseño de Casos de Prueba RF №4

Descripción general					
El sistema debe permitir editar los nombres de las columnas					
SC 1 Editar columna.					
Escenario	Descripción	ColumnName	Typeofcolumn	Respuesta del sistema	Flujo central

EC 1.1 Editar tablas.	Selecciona la pestaña Rename Table la cual muestra una sección para editar la tabla en la base de datos.	Designatio	Integer	Se edita una tabla con los campos especificados.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Edit Column / botón editar.
EC 1.2 Editar tablas presionando el botón cancelar.	Se cancela la opción para editar una tabla.	Designation	Integer	No se edita la tabla con los campos especificados	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Edit Column/ botón cancelar.
EC 1.3 Editar tabla con campo Column Name vacío.	Selecciona la opción dejando campos vacíos.	(vacío)	Integer	Detecta que el campo Table Name está vacío, muestra un mensaje de error y no se edita la columna.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Edit Column/ botón editar.

Tabla A.3. Diseño de Casos de Prueba RF №5

Descripción general			
El sistema debe permitir eliminar los datos de las tablas(filas/columnas)			
SC 1 Eliminar datos en las tablas			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Eliminar tablas.	Selecciona la pestaña Remove Table la cual elimina la tabla de la base de datos.	Se elimina una tabla de la base de datos.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/Manager/ Remove Table/se elimina la tabla automáticamente.

Tabla A.4. Diseño de Casos de Prueba RF №6

Descripción general							
El sistema debe permitir la comunicación (conexión) entre las interfaces de la aplicación.							
SC 1 Realizar la comunicación entre las interfaces.							
Escenario	Descripción	Host	DBName	Username	Pass	Respuesta del sistema	Flujo central

EC 1.1 Establecer la conexión.	Se selecciona el botón Test Connection en la aplicación, si se puede acceder a la BD la conexión es satisfactoria.	localhost	BaseDatos	Admin	admin	Muestra un mensaje estableciendo la conexión de la base de datos con la aplicación.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Test Connection/.
EC 1.2 No se establece la conexión.	Se selecciona el botón Test Connection en la aplicación, no se puede acceder a la BD.	localhost	BaseDatos	(vacío)	admin	Muestra un mensaje de que no hay conexión con la base de datos porque existen campos vacíos .	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Test Connection/.
EC 1.3 Establecer la conexión con campos vacíos.	Se selecciona el botón Test Connection en la aplicación, dejando campos vacíos	localhost	(vacío)	Invitado	(vacío)	Detecta que existen campos vacíos y muestra un mensaje de que no hay conexión con la base de datos porque existen campos vacíos .	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Test Connection/.

Tabla A.5. Diseño de Casos de Prueba RF №7

Descripción general							
El sistema debe permitir la comunicación de la BD mediante un servidor local o remoto.							
SC 1 Permitir la comunicación entre la aplicación del proyecto.							
Escenario	Descripción	Host	DBName	Username	Pass	Respuesta sistema	Flujo central
EC 1.1 Establecer conexión local.	Se accede a la interfaz donde se inserta un host local.	127.0.0.1	BaseDatos	Admin	admin	El sistema debe permitir el acceso a la base de datos desde un host local.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/ DataBase Manger/ Manager/Host.

EC 1.2 Establecer conexión remota.	Se accede a la computadora donde se encuentra la aplicación por un host establecido por la red.	10.8.145.25	BaseDatos	Admin	admin	El sistema debe permitir el acceso a la Base de datos, desde cualquier host.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/ DataBase Manger/ Manager/Host.
EC 1.3 Establecer la conexión con campos vacíos.	Se selecciona el botón Test Connection en la aplicación, dejando campos vacíos	127.0.0.1	BaseDatos	Admin	(vacío)	Detecta que existen campos vacíos y muestra un mensaje de error existen campos vacíos .	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Host.

Tabla A.6. Diseño de Casos de Prueba RF №8

Descripción general				
La BD debe garantizar la estructura de componentes normalizados.				
SC 1 Adicionar componentes o piezas normalizadas.				
Escenario	Descripción	TableName	Respuesta del sistema	Flujo central
EC 1.1 Adicionar componentes normalizados.	Se adiciona en la base de datos solo componentes normalizados.	AS 1427-Metric.	Se adiciona el componente a la base de datos.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager/ Add Table/ botón aceptar.
EC 1.2 Adicionar componente presionando el botón cancelar.	Se cancela la operación para adicionar el componente.	AS 1427-Metric.	No se adiciona el componente.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager Add Table/botón cancelar.
EC 1.3 Adicionar componente con campo Table Name vacío.	Se intenta adicionar una tabla dejando campos vacíos.	(vacío).	Detecta que el campo Table Name está vacío, muestra un mensaje de error y no se adiciona la tabla.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager/ Add Table/botón aceptar.

Tabla A.7. Diseño de Casos de Prueba RF №9

Descripción general

Las tablas de la base de datos deben estar compuestas de filas y columnas.			
SC 1 El sistema debe permitir que todas las tablas estén compuestas por filas y columnas.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Insertar tablas en la BD.	Se selecciona el botón para insertar o adicionar las tablas a la BD.	Se insertan las tablas con sus respectivas filas y columnas.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager Add Table.

Tabla A.8. Diseño de Casos de Prueba RF №10

Descripción general			
Los datos de la BD están clasificados de acuerdo a las normas estándar.			
SC 1 El sistema debe permitir que los datos de las tablas estén estandarizados.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Insertar datos en las tablas por el rol Administrador (Admin).	Se selecciona el botón para insertar o adicionar las tablas a la base de datos.	Se insertan las tablas con los datos estándar.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/Manager Add Table.

Tabla A.9. Diseño de Casos de Prueba RF №11

Descripción general			
La BD debe permitir exportar las tablas en diferentes formatos.			
SC 1 El sistema debe permitir que los datos de todas las tablas puedan ser exportados en diferentes formatos.			
Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Exportar datos de las tablas en formato HTML.	Se selecciona el botón Export y el formato deseado para exportar la tabla.	El sistema permite que la tabla sea exportada en formato HTML.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/ Manager, al presionar el botón Export se despliega una serie de formato y se selecciona el HTML y se guarda el archivo.
EC 1.2 Exportar datos de las tablas en formato ODF.	Se selecciona el botón Export y el formato deseado para exportar la tabla.	El sistema permite que la tabla sea exportada en formato ODF.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/ Manager, al presionar el botón Export se despliega una serie de formato y se selecciona el ODF y se guarda el archivo.

EC 1.3 Exportar datos de las tablas en CVS.	Se selecciona el botón Export y el formato deseado para exportar la tabla.	El sistema permite que la tabla sea exportada en formato CVS.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/ Manager, al presionar el botón Export se despliega una serie de formato y se selecciona el CVS y se guarda el archivo.
EC 1.4 Exportar datos de las tablas en Xlsx.	Se selecciona el botón Export y el formato deseado para exportar la tabla.	El sistema permite que la tabla sea exportada en formato Xlsx.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/DataBase Manger/ Manager, al presionar el botón Export se despliega una serie de formato y se selecciona el Xlsx y se guarda el archivo.

Tabla A.10. Diseño de Casos de Prueba RF №12

Descripción general							
El sistema debe permitir que la BD sea creada dependiendo del rol.							
SC 1 Acceder a los permisos según el rol.							
Escenario	Descripción	Host	DBName	Username	Pass	Respuesta sistema	Flujo central
EC 1.1 Establecer permiso de Admin para crear la base de datos.	Se accede a la aplicación donde se registra como administrador, estableciendo todos los permisos para crear la BD.	localhost	BaseDatos	Admin	admin	El sistema debe permitir el acceso a toda la aplicación.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/User/ Create DataBase.
EC 1.2 Establecer permiso de Invitado.	Se accede a la aplicación donde se registra como invitado, estableciendo solo los permisos para ver las tablas.	localhost	BaseDatos	Invitado	(vacío)	El sistema debe permitir el acceso a visualizar las tablas.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/User/ Invitado.

EC 1.3 Crear base de datos con campos vacíos.	Se accede a la aplicación donde se registra como administrador, estableciendo todos los permisos para crear la BD dejando campos vacíos	(vacío)	BaseDatos	Admin	admin	Detecta que existen campos vacíos y muestra un mensaje de error existen campos vacíos .	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators /DataBase Manger/ Manager/Create DataBase.

Tabla A.11. Diseño de Casos de Prueba RF №13

Descripción general							
El sistema debe permitir que la BD sea eliminada dependiendo del rol.							
SC 1 Acceder a los permisos según el rol.							
Escenario	Descripción	Host	DBName	Username	Pass	Respuesta sistema	Flujo central
EC 1.1 Eliminar la base de datos.	Se accede a la aplicación donde se registra como administrador, estableciendo todos los permisos para eliminar la BD.	localhost	BaseDatos	Admin	admin	El sistema debe permitir eliminar la base de datos.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/ DataBase Manger/ Manager/ Remove DataBase.
EC 1.2 Eliminar la base de datos presionando el botón cancelar.	Se cancela la operación para eliminar la base de datos.	localhost	BaseDatos	Admin	admin	El sistema muestra un mensaje preguntando si desea eliminar la base de datos.	El sistema muestra la interfaz Ingeniero, se presiona el botón Accelerators/ DataBase Manger/ Manager/ Remove DataBase/ botón cancelar.

Tabla A.12. Diseño de Casos de Prueba RF №14

Descripción general
La BD debe permitir consultas, reportes y formularios.
SC 1 El sistema debe permitir que se obtenga respuestas mediante consultas, reportes y formularios.

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Realizar consultas, reportes o formularios.	Se realiza una consulta a la base de datos.	El sistema permite que la consulta sea procesada y devuelve una respuesta.	Manager /databasemanager

A.2. Manual de usuario Soluciones Informáticas para la Ingeniería y la Industria 1.0

A.2.1. Configuración Inicial

Una vez que el cliente tenga instalado el sistema CAD "Ingeniero", deberá instalar el servidor de base de datos PostgresSQL 9.4 y cargar el archivo BD1.backup (base de datos). Luego podrá acceder dentro de la sección *Standard* a la sección de aceleradores de diseño *Accelerators*. Al activarse el botón *DataBase-Manager* se muestra una ventana con diferentes opciones que permite establecer la configuración para la conexión de la base de datos ya creada con el nombre "BaseDatos".

- Opción Host: el usuario puede insertar un host local (127.0.0.1) o remoto.
- Opción *DataBase name*: nombre de la base de datos.
- Opción *username*: nombre del usuario (Admin o Invitado).
- Opción *password*: contraseña asociada al rol establecido.
- Opción *Search DataBase*: contiene las base de datos disponibles, son reflejadas en la opción DataBase name.
- Opción *Test Connection*: permite establecer o no la conexión con la base de datos.
- Opción *Create DataBase*: permite crear una nueva base de datos.
- Opción *Remove DataBase*: permite eliminar una base de datos ya creada.

A.2.2. Servidor remoto

El servidor remoto permite el acceso a la base de datos mediante el protocolo TCP/IP. El usuario en la opción del host decide insertar una dirección ip remota debe configurar en el gestor de base de datos PostgreSQL el fichero postgresql.conf, en este fichero se cambian los parámetros de configuración que afectan al funcionamiento de PostgreSQL en nuestra máquina (listen_addresses = *) y el fichero pg_hba.conf, que permite controlar el acceso al servidor de base de datos (se adiciona uno nuevo insertando type = host, database = all, user = all, ip Address = 0.0.0.0/0 y method = md5).

A.2.3. Seguridad y Control de Acceso

Las personas que trabajan con el sistema deben acceder para autenticarse, utilizando el usuario Admin y password admin, para realizar cualquier operación válida. Al ser un usuario externo al grupo de investigación

deberá acceder con el usuario Invitado y solo podrá visualizar la información contenida en las tablas. Cuando el usuario introduzca una contraseña o usuario inválido, o deje algún campo vacío, el sistema muestra un mensaje de error.

A.2.4. Subsistemas o Módulos

Una vez que el usuario Admin accede al sistema se muestra la vista principal del componente con el nombre *Manager*, esta interfaz cuenta con un conjunto de funcionalidades en el borde superior izquierdo.

- *Add Table*: se adiciona la tabla a la base de datos.
- *Remove Table, Remove Row y Remove Column*: elimina automáticamente de la base de datos la tabla, fila o columna seleccionada.
- *Rename Table*: permite insertar un nuevo nombre a la tabla seleccionada.
- *Add Row*: adiciona la fila al final de la tabla, con su id correspondiente.
- *Add Column*: adiciona la columna con el tipo de dato correspondiente a la tabla.
- *Edit Column*: permite insertar un nuevo nombre a la columna seleccionada con el tipo de dato correspondiente a la dicha columna.
- *Export*: permite exportar los datos de las tablas en formatos como HTML, CVS, ODF y Xlsx.
- La flecha indica regresar a la configuración inicial.

En el extremo derecho de la interfaz se visualizan las tablas contenidas en la base de datos, al presionar una de ellas podrá observar la información almacenada en la misma.