

## **Actividad | 1 | Creación de la Base de Datos**

### **Lenguajes de Programación II**

---

Ingeniería en Desarrollo de  
Software



**TUTOR: Felix Acosta Hernández**

**ALUMNO: Sarahi Jaqueline Gómez Juárez, sara\_2mil@outlook.com**

**FECHA: viernes, 02 de agosto de 2024.**

## Índice

<b>Índice.....</b>	<b>2</b>
<b>Introducción .....</b>	<b>4</b>
<b>Descripción .....</b>	<b>9</b>
<b>Justificación .....</b>	<b>17</b>
<b>Desarrollo.....</b>	<b>24</b>
<b>Contextualización: .....</b>	<b>24</b>
<i>Datos internos que debe tener la Tabla Centro_de_Trabajo: .....</i>	<i>25</i>
<i>Ejemplo de Los Datos que deberán aparecer en la Consulta.....</i>	<i>26</i>
<i>Modelo Entidad-Relación:.....</i>	<i>27</i>
<i>Modelo Lógico Relacional.....</i>	<i>30</i>
<b>Base de Datos (BD) en Microsoft SQL Server para el sistema:.....</b>	<b>34</b>
<i>Gestor de Bases de Datos SQL Server: .....</i>	<i>34</i>
<i>Paso 1: Creación de la Base de Datos y la tabla Centro_de_Trabajo (Base de datos, parte 1):.....</i>	<i>35</i>
<i>Paso 2: Creación de la tabla Empleado (Base de datos, parte 2): .....</i>	<i>35</i>
<i>Paso 3: Creación de la tabla Directivo (Base de datos, parte 3): .....</i>	<i>36</i>
<i>Confirmación de la creación de la tabla Centro_de_Trabajo:.....</i>	<i>36</i>
<i>Confirmación de la creación de la tabla Directivo:.....</i>	<i>37</i>
<i>Confirmación de la creación de la tabla EMPLEADO:.....</i>	<i>37</i>
<i>Insertar datos en la tabla Centro_de_Trabajo:.....</i>	<i>38</i>
<i>Consulta que confirma que se han insertado correctamente los datos en la tabla Centro_de_trabajo: .....</i>	<i>38</i>

<i>Insertar datos en la tabla EMPLEADO parte 1:</i> .....	39
<i>Insertar datos en la tabla EMPLEADO parte 2:</i> .....	39
<i>Consulta que confirma que se han insertado correctamente los datos en la tabla EMPLEADO parte 1:</i> .....	40
<i>Consulta que confirma que se han insertado correctamente los datos en la tabla EMPLEADO parte 2:</i> .....	40
<i>Insertar datos en la tabla Directivo:</i> .....	41
<i>Consulta que confirma que se han insertado correctamente los datos en la tabla Directivo:</i> .....	41
<i>Consulta para verificar si es o no Directivo el empleado (parte 1):</i> .....	42
<i>Consulta para verificar si es o no Directivo el empleado (parte 2):</i> .....	44
<i>Consulta para verificar si es o no Directivo el empleado (parte 3):</i> .....	44
<b>Conclusión:</b> .....	45
<b>Referencias</b> .....	51

## Introducción

El presente proyecto aborda el análisis exhaustivo de los datos presentados en la sección "**La Contextualización**" con el propósito de diseñar y desarrollar una Base de Datos robusta en SQL Server, asimismo, se exponen el **Modelo Entidad-Relación** y el **Modelo Lógico Relacional**, facilitando una comprensión profunda del sistema que se está construyendo.

El proyecto se estructura en tres etapas, y en esta primera fase nos enfocaremos en la cimentación conceptual y técnica, este enfoque inicial sentará las bases necesarias para la posterior implementación de un programa en lenguaje C++, que integrará y operará sobre la Base de Datos diseñada.

Es crucial recordar que un lenguaje de programación consiste en un conjunto de reglas y convenciones que permiten al programador redactar instrucciones comprensibles para la computadora, estas instrucciones, conocidas como código fuente, están diseñadas de tal manera que pueden ser traducidas por un compilador o un intérprete, dependiendo del tipo de lenguaje utilizado, existen varios tipos de lenguajes de programación, cada uno diseñado para propósitos específicos y con diferentes niveles de abstracción, por ejemplo: C++: Es una extensión de C que incorpora características de programación orientada a objetos.

Una **Base de Datos** es un conjunto organizado de datos almacenados y accesibles electrónicamente, estas colecciones de datos están estructuradas de manera que permiten su fácil acceso, manipulación y actualización, las Bases de Datos son fundamentales en aplicaciones informáticas, ya que permiten almacenar información de manera eficiente y realizar operaciones complejas para gestionar grandes volúmenes de datos, al ser ejecutada en **SQL Server** que es un gestor de Bases de Datos, mejor definido como un Sistema de Gestión de Bases de Datos Relacionales (RDBMS, por sus siglas en inglés), un RDBMS como SQL Server permite a los

usuarios crear, gestionar, y manipular Bases de Datos relacionales, donde los datos se organizan en tablas que se relacionan entre sí a través de claves primarias y foráneas.

SQL Server proporciona herramientas y servicios para realizar operaciones sobre los datos, tales como consultas, actualizaciones, inserciones y eliminaciones, además de ofrecer funciones avanzadas de seguridad, escalabilidad, rendimiento y recuperación ante desastres, por lo tanto, sí, SQL Server actúa como un Gestor de Bases de Datos al facilitar el almacenamiento, la administración y la recuperación de datos de manera eficiente y segura.

Pero para ello, es fundamental primero realizar los modelos antes mencionados, el **Modelo Entidad-Relación (ER)** es una herramienta de modelado conceptual creada por Peter Chen en 1976, fundamental en el diseño de Bases de Datos, este método se utiliza para representar visualmente los datos de un sistema y las relaciones entre ellos, se basa en tres componentes clave: las entidades (objetos o conceptos relevantes), sus atributos (características) y las relaciones (asociaciones) entre las entidades, al proporcionar una representación gráfica y abstracta, el modelo ER facilita la estructuración y organización lógica de los datos antes de su implementación en una base de datos, por su parte el **Modelo Lógico Relacional** es una representación detallada de la estructura de una Base de Datos que especifica cómo se organizan y relacionan las tablas, columnas y claves en un sistema de gestión de Bases de Datos relacional. Este modelo se sitúa entre el diseño conceptual (como el Modelo Entidad-Relación) y la implementación física en un sistema de gestión de Bases de Datos (RDBMS), en este modelo se centra en definir las tablas, sus atributos, las claves primarias, y las relaciones entre tablas para asegurar la integridad y de manera que se puedan implementar directamente en un sistema de Base de Datos relacional.

Algunos detalles de este modelo:

**Tablas:** Representan entidades y contienen datos organizados en filas y columnas.

**Columnas:** Definen los atributos de las entidades.

**Filas:** Cada fila representa una instancia única de la entidad.

**Claves Primarias:** Identifican de manera única cada registro en una tabla.

**Claves Foráneas:** Establecen relaciones entre tablas.

**Restricciones:** Aseguran la integridad de los datos y las reglas del negocio.

**Normalización:** A menudo, el modelo lógico relacional incorpora procesos de normalización para reducir la redundancia de datos y evitar anomalías en las actualizaciones, este proceso implica dividir las tablas en tablas más pequeñas y establecer relaciones entre ellas.

**Tipos de Datos:** En el modelo lógico relacional, se especifican los tipos de datos de las columnas para garantizar que los datos almacenados sean consistentes y válidos, por ejemplo, una columna puede estar definida para almacenar solo enteros, texto, fechas, etc.

**Índices:** Aunque más relacionado con el diseño físico, los índices se mencionan a veces en el contexto lógico para optimizar el rendimiento de las consultas al permitir un acceso más rápido a los datos.

**Integridad Referencial:** Es fundamental mantener la integridad referencial a través de las claves foráneas, asegurando que las relaciones entre tablas sean válidas y que los datos en una tabla se correspondan con los datos en otra.

**Dependencias Funcionales:** En el modelo lógico relacional, es importante identificar las dependencias funcionales para asegurarse de que la normalización se realice correctamente y que las relaciones entre las tablas reflejen adecuadamente las dependencias entre los datos.

**Integridad de los Datos:** Además de las claves primarias y foráneas, el modelo lógico puede definir otras restricciones para mantener la integridad de los datos, como restricciones de

unicidad, restricciones de dominio y restricciones de valores nulos.

**Esquema:** El modelo lógico también define el esquema de la base de datos, que incluye la definición de las tablas, columnas, y las relaciones entre ellas, este esquema es esencial para la creación de la Base de Datos en el RDBMS.

**Consultas y Vistas:** Aunque más asociado con el diseño físico, en algunos casos el modelo lógico puede incluir la definición de vistas o consultas predefinidas que simplifiquen el acceso a los datos para los usuarios finales.

**El Modelo Lógico Relacional** facilita la implementación del diseño conceptual en una Base de Datos funcional, garantizando que la estructura de datos sea eficiente y coherente.

Estos conceptos utilizan la **Programación Orientada a Objetos (POO)** que es un paradigma de programación que organiza el código en torno a objetos, que son instancias de clases, las clases definen las propiedades (atributos) y comportamientos (métodos) de los objetos, la POO se basa en conceptos como encapsulamiento (ocultación de datos), herencia (reutilización de código), polimorfismo (múltiples formas de métodos) y abstracción (simplificación del diseño), dicho en otras palabras la programación orientada a objetos (POO) es un enfoque de programación que utiliza "objetos" para representar datos y comportamientos, los objetos son instancias de "clases", que definen un conjunto de atributos (datos) y métodos (funciones) asociados, los principales conceptos de la POO son:

**Encapsulamiento:** Oculta los detalles internos y solo expone lo necesario para interactuar con el objeto.

**Herencia:** Permite a una clase derivada heredar atributos y métodos de una clase base.

**Polimorfismo:** Permite que métodos con el mismo nombre se comporten de manera diferente según el objeto que los llama.

**Abstracción:** Simplifica la complejidad al enfocarse en las características esenciales de un objeto y ocultar detalles innecesarios

La programación orientada a objetos suele utilizar mucho los **constructores** y **destructores** son métodos especiales de una clase que gestionan la creación y destrucción de objetos.

Un **constructor** es un método especial que se invoca automáticamente cuando se crea un nuevo objeto de una clase, su propósito principal es inicializar los atributos del objeto con valores predeterminados o proporcionados, los constructores tienen el mismo nombre que la clase y no tienen un tipo de retorno, a diferencia de un **destructor** que es un método especial que se invoca automáticamente cuando un objeto está a punto de ser destruido o eliminado de la memoria, su función principal es realizar tareas de limpieza y liberar recursos que el objeto haya utilizado, como cerrar archivos o liberar memoria.

Finalmente, identificaremos la importancia de adquirir este conocimiento en nuestra vida cotidiana y laboral, recordando las palabras de Michael J. Hernandez: “El diseño de una Base de Datos debe ser como la arquitectura de un edificio: estructurado, funcional y estéticamente agradable.”



## Descripción

En el presente documento podrás visualizar la Base de Datos desarrollada en SQL Server y dos distintos diagramas conocidos como el Modelo Entidad-Relación y el Modelo Lógico-Relacional, con la finalidad de desarrollar el sistema solicitado durante la sección de “**La Contextualización**”.

Una **Base de Datos** es un sistema organizado para almacenar, gestionar y recuperar información de manera eficiente, es una colección de datos relacionados que están estructurados para facilitar su acceso, manipulación y actualización, algunos tipos de Bases de Datos:

**Bases de Datos Relacionales (RDBMS):** Utilizan tablas para organizar datos y relaciones entre ellos mediante claves primarias y foráneas, SQL (Structured Query Language) es el lenguaje estándar para interactuar con estas Bases de Datos, por ejemplo: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

**Bases de Datos NoSQL:** Diseñadas para manejar grandes volúmenes de datos no estructurados o semi-estructurados, pueden ser documentales, clave-valor, columnares o de grafos, por ejemplo: MongoDB, Cassandra, Redis, CouchDB.

**Bases de Datos Orientadas a Objetos:** Integran conceptos de programación orientada a objetos y permiten almacenar objetos directamente en la base de datos, por ejemplo: db4o, ObjectDB.

**Bases de Datos en la Nube:** Ofrecen servicios de Bases de Datos como un servicio (DBaaS), facilitando la gestión y escalabilidad sin necesidad de infraestructura local, por ejemplo: Amazon RDS, Google Cloud SQL, Azure SQL Database.

El diseño de una Base de Datos implica varios pasos cruciales:

**Modelado de Datos:** Crear un modelo conceptual (ERD - Diagrama de Entidad-Relación y el diagrama Lógico Racional) que defina entidades, relaciones y atributos, estos serán explicados más adelante.

**Normalización:** Organizar los datos para reducir la redundancia y mejorar la integridad, esto implica dividir las tablas en estructuras más pequeñas y relacionadas.

**Implementación:** Crear las estructuras físicas en el sistema de gestión de Bases de Datos elegido, como tablas, índices y vistas.

**Mantenimiento:** Asegurarse de que la Base de Datos se mantenga eficiente y actualizada con el tiempo, realizando tareas de respaldo, optimización y actualización

**Un Diagrama Entidad-Relación** (ERD, por sus siglas en inglés) es una representación gráfica utilizada en la ingeniería de software y en la administración de Bases de Datos para modelar la estructura lógica de un sistema de información, este diagrama ilustra las entidades del sistema, sus atributos y las relaciones entre ellas.

#### **Componentes principales de un ERD:**

**Entidades:** Representan objetos o cosas del mundo real que tienen existencia independiente en la base de datos. Se dibujan como rectángulos y pueden ser, por ejemplo, "Director", "EMPLEADO", o "Centro\_de\_Trabajo".

**Atributos:** Son las propiedades o características de las entidades. Se representan como óvalos conectados a la entidad correspondiente, por ejemplo, un "EMPLEADO" podría tener atributos como "Nombre\_Empleado", "Apellido\_Paterno", y "Apellido\_Materno".

**Relaciones:** Describen cómo las entidades están conectadas entre sí, se dibujan como rombos y se conectan a las entidades relacionadas, por ejemplo, una relación entre "Empleados" y "Directivo" podría indicar que un empleado puede ser directivo, en cuyo caso se gestiona con

la relación "ES UN".

**Cardinalidad:** Indica el número de ocurrencias de una entidad que puede estar asociada con las ocurrencias de otra entidad en una relación, se expresa comúnmente como 1:1, 1, o M.

Por otro lado, **Un Diagrama Lógico-Relacional**, también conocido como diagrama Relacional, representa la estructura lógica de una Base de Datos en el modelo relacional, este diagrama muestra cómo las tablas (o relaciones) en una Base de Datos relacional están conectadas y cómo se estructuran los datos en términos de claves primarias, claves foráneas y otros atributos.

#### **Componentes principales de un Diagrama Lógico-Relacional:**

**Tablas (Relaciones):** Representan conjuntos de datos organizados en filas y columnas. Cada tabla se dibuja como un rectángulo y contiene atributos (columnas) que describen las propiedades de los datos. Por ejemplo, una tabla llamada "EMPLEADOS" podría tener columnas como "ID\_Empleado", "Nombre", y "Apellido\_Paterno" entre otras.

**Atributos (Columns):** Dentro de cada tabla, los atributos representan las características o propiedades de los datos almacenados en la tabla, cada atributo se asocia a un tipo de datos (por ejemplo, entero, texto, fecha) y puede incluir restricciones como "NO NULO" o "ÚNICO".

**Llaves Primarias (Primary Keys):** Son atributos o combinaciones de atributos que identifican de manera única cada fila en una tabla, en el diagrama, una llave primaria se suele subrayar o destacar. Por ejemplo, en la tabla " EMPLEADO ", " ID\_Empleado " podría ser la llave primaria.

**Llaves Foráneas (Foreign Keys):** Son atributos en una tabla que crean una relación con otra tabla, las llaves foráneas apuntan a las llaves primarias de otras tablas y permiten establecer relaciones entre los datos, estas se suelen indicar mediante líneas que conectan las tablas

correspondientes.

**Relaciones:** Representan la conexión entre diferentes tablas a través de llaves foráneas, estas conexiones se ilustran mediante líneas que unen las tablas, y pueden tener diferentes tipos de cardinalidad (1:1, 1, N).

**Relación Uno a Uno (1:1):** En una relación uno a uno, un registro en la primera tabla está asociado con un solo registro en la segunda tabla, y viceversa.

**Relación Uno a Muchos (1:M):** En una relación uno a muchos, un registro en la primera tabla está asociado con varios registros en la segunda tabla, pero cada registro en la segunda tabla está asociado con solo un registro en la primera tabla.

**Relación Muchos a Muchos (N:M):** En una relación muchos a muchos, varios registros en la primera tabla pueden estar asociados con varios registros en la segunda tabla, para implementar esto, se necesita una tabla intermedia (o de unión) que contenga claves foráneas de ambas tablas.

**Relación Jerárquica:** Una relación jerárquica es un tipo especial de relación uno a muchos donde los registros están organizados en una estructura de árbol o jerarquía.

Estas relaciones ayudan a definir cómo los datos están conectados y permiten realizar consultas complejas que extraen y manipulan datos en función de esas conexiones.

En este mismo también se definen los tipos de datos que se utilizarán en SQL Server, ya que en SQL Server, cada columna en una tabla tiene un tipo de dato asociado que define qué tipo de datos puede almacenar, algunos tipos comunes son:

**Tipo Numéricos:**

**INT:** Enteros.

**FLOAT:** Números en punto flotante.

**DECIMAL:** Números decimales con precisión definida.

**Tipo Texto:**

**VARCHAR(n):** Cadenas de texto de longitud variable, donde n es el tamaño máximo.

**NVARCHAR(n):** Cadenas de texto Unicode de longitud variable, útil para soportar caracteres internacionales.

**Tipo Fecha y Hora:**

**DATE:** Solo fecha.

**DATETIME:** Fecha y hora.

**DATETIME2:** Fecha y hora con mayor precisión que DATETIME.

**Tipo Booleano:**

**BIT:** Almacena valores de verdadero (1) o falso (0).

**Tipo Binarios:**

**VARBINARY(n):** Datos binarios de longitud variable.

**IMAGE:** Datos binarios grandes, como imágenes.

**Otros tipos:**

**TEXT:** Texto largo, obsoleto en favor de VARCHAR(MAX).

**NTEXT:** Texto largo Unicode, obsoleto en favor de NVARCHAR(MAX).

Cada tipo de dato tiene sus propias características y restricciones, y elegir el tipo adecuado es importante para el rendimiento y la integridad de los datos.

Ambos utilizan algunos conceptos básicos de la programación orientada a objetos (POO):

La **Programación Orientada a Objetos (POO)** es un paradigma de programación que organiza el software en torno a "objetos" en lugar de "funciones" o "lógicas de programación" tradicionales, algunos conceptos básicos son:

**Clases y Objetos:**

**Clase:** Es una plantilla o modelo para crear objetos, define un tipo de objeto con sus propiedades (atributos) y comportamientos (métodos).

**Objeto:** Es una instancia de una clase, cada objeto tiene su propio estado y puede ejecutar los métodos definidos en su clase.

**Encapsulamiento:** Se refiere a la práctica de ocultar el estado interno del objeto y requerir que toda interacción se realice a través de métodos, esto ayuda a proteger los datos y a mejorar la modularidad del código.

**Herencia:** Permite a una clase derivada (o subclase) heredar atributos y métodos de una clase base (o superclase), la herencia facilita la reutilización de código y la creación de una jerarquía de clases, existen distintos tipos de herencia solo por mencionar algunos:

**Herencia Simple:** Una clase derivada (o subclase) hereda de una sola clase base (o superclase).

**Herencia Múltiple:** Una clase puede heredar de múltiples clases base, este tipo de herencia no es soportado directamente por todos los lenguajes de programación debido a problemas como la ambigüedad del diamante (cuando dos clases base tienen un método común, la clase derivada puede no saber cuál método utilizar).

**Herencia Jerárquica:** Una sola clase base es heredada por múltiples clases derivadas, cada subclase puede tener sus propias características adicionales.

**Herencia Multinivel:** Una clase hereda de una clase que ya es una subclase de otra clase. Se forma una cadena de herencia.

**Herencia de Interfaces:** Una clase implementa uno o más interfaces, las interfaces definen métodos que la clase debe implementar, pero no proporcionan la implementación.

**Herencia de Implementación (Mixins):** Una técnica para añadir funcionalidad a una clase mediante la combinación de varias clases base que proporcionan implementaciones adicionales, no es un tipo de herencia formal en todos los lenguajes, pero es común en lenguajes que soportan herencia múltiple o mixins.

Cada tipo de herencia ofrece ventajas y desventajas dependiendo de la complejidad del diseño del software y los requisitos específicos, la elección del tipo de herencia debe basarse en la claridad del modelo y la facilidad de mantenimiento del código.

### **Polimorfismo:**

Es la capacidad de una función o método para tomar múltiples formas, en la POO, esto significa que un método puede comportarse de diferentes maneras según el objeto que lo llama, a través de la sobrecarga de métodos o la implementación de métodos en clases derivadas.

### **Abstracción:**

Consiste en simplificar un sistema al enfocarse en sus características esenciales y ocultar los detalles complejos, las clases abstractas y las interfaces son herramientas que se utilizan para implementar la abstracción en la POO.

### **Constructores y Destructores:**

**Constructor:** Es un método especial que se llama cuando se crea un objeto, se usa para inicializar el objeto, estos no se heredan automáticamente, cada clase derivada debe definir sus propios constructores y puede invocar los constructores de la clase base explícitamente.

**Destructor:** Es un método que se llama cuando un objeto se destruye, y se utiliza para liberar recursos o realizar otras tareas de limpieza. Estos en lenguajes como C++, los destructores de la clase base se llaman automáticamente al destruir un objeto derivado, en Java y Python, el manejo de recursos y la limpieza pueden requerir enfoques específicos del lenguaje.

Métodos y Atributos:

Métodos: Son funciones definidas dentro de una clase que definen los comportamientos del objeto.

Atributos: Son variables definidas dentro de una clase que representan el estado o las características del objeto.

La **Programación Orientada a Objetos** es un paradigma extenso y profundo que ofrece numerosas características y técnicas avanzada, su comprensión facilita el aprendizaje de lenguajes de programación, que son conjuntos de reglas y sintaxis que permiten a los programadores escribir instrucciones que las computadoras pueden interpretar y ejecutar, estos lenguajes se utilizan para desarrollar software, aplicaciones y sistemas operativos, entre otros, y cada uno tiene sus propias reglas, estructuras y paradigmas.

Los lenguajes de programación se pueden clasificar en varias categorías:

Lenguajes de bajo nivel: Cercanos al lenguaje de máquina, como el ensamblador. Son eficientes pero menos intuitivos.

Lenguajes de alto nivel: Más abstractos y fáciles de usar, como Python, Java y C++. Estos permiten trabajar con conceptos más cercanos al lenguaje humano.

Lenguajes interpretados: Se ejecutan línea por línea, como Python y JavaScript.

Lenguajes compilados: Se traducen completamente a código de máquina antes de la ejecución, como C y C++.

Cada tipo de lenguaje tiene sus ventajas y desventajas dependiendo del contexto en el que se utilice.

“El diseño de una Base de Datos es una forma de resolver problemas mediante la organización sistemática de la información.” - Peter Rob y Carlos Coronel



### **Justificación**

El objetivo de este proyecto es la implementación de una Base de Datos robusta y bien estructurada, el diseñar y desarrollar una Base de Datos utilizando SQL Server, ofrece múltiples beneficios tanto académicos como profesionales, permitiendo a la aplicación de los conceptos teóricos aprendidos en el curso "Lenguajes de Programación II", específicamente en el ámbito de las Bases de Datos relacionales, a través de la práctica, se consolidan conocimientos sobre el diseño y gestión de Bases de Datos, lo cual es esencial para cualquier ingeniero de software.

Las Bases de Datos es un componente crucial en el desarrollo de aplicaciones en diversas industrias, dominar herramientas como SQL Server y comprender los principios de modelado de datos proporciona una gran ventaja competitiva en el mercado laboral, así mismo si está misma está bien diseñada, garantiza los datos se almacenen de manera eficiente y organizada, facilitando su acceso, manipulación y actualización, esto es vital para el rendimiento y la integridad de cualquier sistema informático, al aplicarla en SQL Server ofrece funciones avanzadas de seguridad y escalabilidad, lo cual es crucial para proteger los datos y manejar grandes volúmenes de información conforme el sistema crece.

#### **Las Bases de Datos ofrecen varias ventajas significativas:**

**Organización y Eficiencia:** Permiten almacenar datos de manera estructurada y organizada, facilitando el acceso y la manipulación eficiente de la información.

**Integridad de Datos:** Mantienen la consistencia, exactitud y validez de los datos a través de reglas y restricciones.

**Seguridad:** Proveen mecanismos para proteger los datos contra accesos no autorizados y pérdidas, a través de permisos y copias de seguridad.

**Reducción de Redundancia:** Minimizan la duplicación de datos mediante la

normalización, lo que ahorra espacio y reduce errores.

**Acceso Concurrente:** Permiten que múltiples usuarios accedan y trabajen con los datos simultáneamente sin conflictos.

**Consultas y Reportes:** Facilitan la generación de consultas complejas y reportes detallados de manera rápida y eficiente.

**Escalabilidad:** Son capaces de crecer y adaptarse a volúmenes de datos más grandes sin perder rendimiento.

**Recuperación de Información:** Permiten recuperar datos fácilmente mediante lenguajes de consulta como SQL.

Estas ventajas hacen que las Bases de Datos sean fundamentales para la gestión y análisis de grandes volúmenes de información en diversas aplicaciones y sectores.

Aplicar la Base de Datos en SQL Server ofrece varias ventajas específicas:

**Rendimiento y Escalabilidad:** SQL Server está diseñado para manejar grandes volúmenes de datos y transacciones de manera eficiente, ofreciendo un rendimiento óptimo incluso bajo carga pesada.

**Seguridad:** SQL Server proporciona avanzadas características de seguridad como autenticación y autorización, encriptación de datos y auditoría, ayudando a proteger la información contra accesos no autorizados y amenazas.

**Alta Disponibilidad:** Ofrece opciones de alta disponibilidad como el clustering, la replicación y los grupos de disponibilidad Always On, asegurando que los datos estén disponibles incluso en caso de fallos del sistema.

**Integración y Compatibilidad:** SQL Server se integra bien con otras herramientas de Microsoft, como Excel, Power BI y Azure, facilitando la importación, exportación y análisis de

datos.

**Herramientas de Administración:** Proporciona un conjunto de herramientas robustas como SQL Server Management Studio (SSMS) y SQL Server Data Tools (SSDT) para la administración, desarrollo y mantenimiento de Bases de Datos.

**Soporte para Transacciones:** Garantiza la consistencia y confiabilidad de los datos mediante el uso de transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).

**Analítica y BI:** Incluye capacidades avanzadas de analítica y business intelligence (BI), como SQL Server Analysis Services (SSAS) y SQL Server Reporting Services (SSRS), para realizar análisis de datos y generar informes detallados.

**Soporte y Comunidad:** Al ser un producto ampliamente utilizado, cuenta con un extenso soporte técnico y una comunidad activa que puede ayudar a resolver problemas y compartir conocimientos.

Estas ventajas hacen que SQL Server sea una opción poderosa y flexible para gestionar Bases de Datos en entornos empresariales y aplicaciones críticas.

El desarrollo de este proyecto brida grandes habilidades prácticas en la creación y gestión de Bases de Datos, ya que para esto incluye la implementación de modelos Entidad-Relación y Lógico-Relacional, la creación de tablas, inserción de datos y realización de consultas.

La implementación de modelos Entidad-Relación (ER) y Lógico-Relacional en Bases de Datos ofrece varias ventajas clave:

Ventajas del **Modelo Entidad-Relación (ER):**

**Claridad y Comunicación:** Facilita la comprensión y comunicación de la estructura de la Base de Datos mediante un diagrama visual, que es fácil de interpretar tanto por técnicos como por no técnicos.

**Diseño Intuitivo:** Proporciona un enfoque intuitivo para el modelado de datos, permitiendo identificar y definir claramente las entidades, relaciones y atributos.

**Prevención de Errores:** Ayuda a identificar y resolver problemas de diseño temprano en el proceso de desarrollo, reduciendo la probabilidad de errores e inconsistencias.

**Documentación:** Sirve como documentación útil para la base de datos, proporcionando una referencia clara para desarrolladores y administradores de Bases de Datos.

#### Ventajas del **Modelo Lógico-Relacional:**

**Normalización y Reducción de Redundancia:** Permite la normalización de datos, lo que reduce la redundancia y mejora la integridad de los datos al dividirlos en tablas relacionadas de manera lógica.

**Flexibilidad:** Facilita la modificación y expansión de la base de datos, permitiendo agregar nuevas tablas o columnas sin afectar significativamente el sistema existente.

**Eficiencia en Consultas:** Optimiza las consultas y operaciones de Base de Datos al organizar los datos en tablas normalizadas, mejorando el rendimiento y la velocidad de acceso a la información.

**Integridad Referencial:** Implementa reglas de integridad referencial, asegurando que las relaciones entre tablas se mantengan coherentes y válidas.

#### **Ventajas Combinadas de Implementar ambos Modelos:**

**Diseño Coherente y Eficiente:** Utilizar ambos modelos proporciona una transición fluida del diseño conceptual al lógico, garantizando que el modelo conceptual (ER) se traduzca correctamente en un modelo lógico-relacional eficiente y funcional.

**Mejora de la Calidad de Datos:** El enfoque sistemático en el diseño y la normalización mejora la calidad y la integridad de los datos, minimizando la duplicación y los errores.

**Facilidad de Mantenimiento:** La estructura clara y bien definida facilita el mantenimiento y la actualización de la base de datos, ya que los cambios se pueden realizar de manera estructurada y controlada.

**Documentación y Seguimiento:** Proporcionan una base sólida para la documentación y el seguimiento de la evolución del diseño de la base de datos, lo cual es esencial para proyectos a largo plazo y colaborativos.

En resumen, la implementación de modelos Entidad-Relación y Lógico-Relacional proporciona un marco estructurado y eficiente para el diseño, implementación y mantenimiento de Bases de Datos, mejorando la claridad, integridad y rendimiento del sistema de gestión de datos.

La posterior implementación de un programa en lenguaje C++ que opere sobre la Base de Datos diseñada permite entender la integración entre la Base de Datos y aplicaciones de software, un conocimiento crucial para el desarrollo de sistemas completos.

Estos conceptos básicos están dentro de la Programación Orientada a Objetos (POO), esta ofrece varias ventajas y es fundamental para el desarrollo de software moderno.

### **Ventajas de la Programación Orientada a Objetos (POO):**

**Modularidad:** Facilita la división de un programa en módulos más pequeños y manejables (clases y objetos), lo que hace que el desarrollo y el mantenimiento del software sean más sencillos.

**Reutilización de Código:** Promueve la reutilización de código mediante la herencia y la creación de bibliotecas de clases reutilizables, lo que reduce el esfuerzo de desarrollo y los errores.

**Abstracción:** Permite crear modelos de objetos del mundo real, proporcionando una

forma intuitiva de estructurar y resolver problemas complejos mediante la abstracción de detalles innecesarios.

**Encapsulamiento:** Protege los datos mediante la encapsulación, lo que impide el acceso no autorizado y promueve la integridad de los datos al controlar cómo se accede y modifica la información.

**Polimorfismo:** Permite a los objetos de diferentes clases ser tratados como objetos de una clase común, facilitando la implementación de funciones genéricas y aumentando la flexibilidad y extensibilidad del código.

**Mantenibilidad:** Hace que el código sea más fácil de mantener y modificar debido a su estructura clara y modular, los cambios en una parte del sistema tienen un impacto mínimo en otras partes.

**Facilidad de Depuración:** Facilita la identificación y corrección de errores, ya que los problemas se pueden aislar dentro de clases y objetos específicos.

Aplicaciones y Beneficios en la Práctica:

**Desarrollo de Software a Gran Escala:** Esencial para el desarrollo de aplicaciones grandes y complejas, como sistemas empresariales, aplicaciones web y software de misión crítica.

**Colaboración en Equipos:** Facilita el trabajo en equipo, ya que los desarrolladores pueden trabajar en diferentes partes del sistema de manera independiente y coherente.

**Patrones de Diseño:** Muchos patrones de diseño (como Singleton, Factory, Observer, etc.) están basados en POO, proporcionando soluciones probadas para problemas comunes en el desarrollo de software.

**Frameworks y Bibliotecas:** La mayoría de los frameworks y bibliotecas modernas

(como Spring, Django, .NET, etc.) están diseñados utilizando principios de POO, por lo que el conocimiento de POO es crucial para utilizarlos eficazmente.

**Adopción de Buenas Prácticas:** Fomenta la adopción de buenas prácticas de programación, como el código limpio, la responsabilidad única y la separación de preocupaciones, que resultan en software más robusto y mantenible.

En resumen, aprender POO proporciona una base sólida para el desarrollo de software de alta calidad, mejorando la modularidad, la reutilización, la mantenibilidad y la colaboración en proyectos de programación.

## Desarrollo

### Contextualización:

Se necesita una estructura de clases que permita a la empresa UNI controlar los distintos tipos de empleados, así como sus datos personales, esto se hará a través de clases, herencia de clases y atributos.

Las clases, por su parte, deberán ser usadas desde una aplicación donde se gestione la siguiente información:

- **Número de Empleado:** (autogenerado, numérico)
- **Nombre:** (capturable, alfanumérico)
- **Apellido Paterno:** (capturable, alfanumérico)
- **Apellido Materno:** (capturable, alfanumérico)
- **Fecha de Nacimiento:** (capturable tipo fecha)
- **RFC:** (calculado conforme al nombre y fecha de nacimiento, alfanumérico)
- **Centro de Trabajo:** (capturable, alfanumérico, elegible desde el número de clave con base en el catálogo de puestos)
- **Puesto:** (capturable, alfanumérico)
- **Descripción del Puesto:** (capturable, alfanumérico)
- **Directivo:** (bandera para indicar tipo de empleado; para directivo 1; para empleado normal 0)

Considerar lo siguiente:

Existe un tipo de empleado denominado Directivo, el cual presenta, además de las cualidades anteriores, atributos particulares de su tipo, los atributos de los directivos son:



- **Número del centro que supervisa** (numérico, capturable)
- **Prestación de combustible** (bandera que indica si el directivo recibe apoyo de combustible)

**Figura 1**

*Datos internos que debe tener la Tabla Centro\_de\_Trabajo:*

**Catálogo de puestos a utilizar:**

Número de Centro	Nombre de Centro	Ciudad
000201	Tiendas Ángel Flores Ropa	Culiacán
000202	Tiendas Ángel Flores Muebles	Culiacán
000203	Tiendas Ángel Flores Cajas	Culiacán
049001	La Primavera Ropa	Culiacán
049002	La Primavera Muebles	Culiacán
049003	La Primavera Cajas	Culiacán

Nota: En la imagen presentada, se muestran los datos que se registrarán en la tabla *ID\_Centro\_de\_trabajo*, esta imagen fue recuperada el 1 de agosto de 2024, de la siguiente fuente: Ag Collage. (s. f.). *Lenguajes de Programación II: Actividad 1 - Creación de la base de datos*. Recuperado de:

[https://agcollege.edu.mx/literaturas/59/15/Actividad\\_1\\_Lenguajes\\_de\\_Programacion\\_II\\_V5.docx.pdf](https://agcollege.edu.mx/literaturas/59/15/Actividad_1_Lenguajes_de_Programacion_II_V5.docx.pdf)

Con todos estos datos previamente capturados, deberá imprimirse un reporte (con el formato que se muestra a continuación): Ejemplo de datos:

**Figura 2**

***Ejemplo de Los Datos que deberán aparecer en la Consulta***

Numero de Empleado	Nombre + Apellido Paterno + Apellido Materno	Fecha de Nacimiento	RFC	Nombre de Centro	Descripción del Puesto	¿Es Directivo?
1	Jesús Vega Castro	26/03/1988	VECJ880326 XXX	Tiendas Ángel Flores Ropa	Vendedor	No
2	José López Pérez	01/01/1980	LOPJ800101 XXX	La Primavera Cajas	Gerente	Si
3	*****	*****	*****	*****	*****	*****

Nota: En la imagen presentada, se muestran los datos que se mostraran en la consulta, esta imagen fue recuperada el 1 de agosto de 2024, de la siguiente fuente: Ag Collage. (s. f.).

*Lenguajes de Programación II: Actividad 1 - Creación de la base de datos.* Recuperado de:

[https://agcollege.edu.mx/literaturas/59/15/Actividad\\_1\\_Lenguajes\\_de\\_Programacion\\_II\\_V5.docx.pdf](https://agcollege.edu.mx/literaturas/59/15/Actividad_1_Lenguajes_de_Programacion_II_V5.docx.pdf)

**Actividad: 1.**

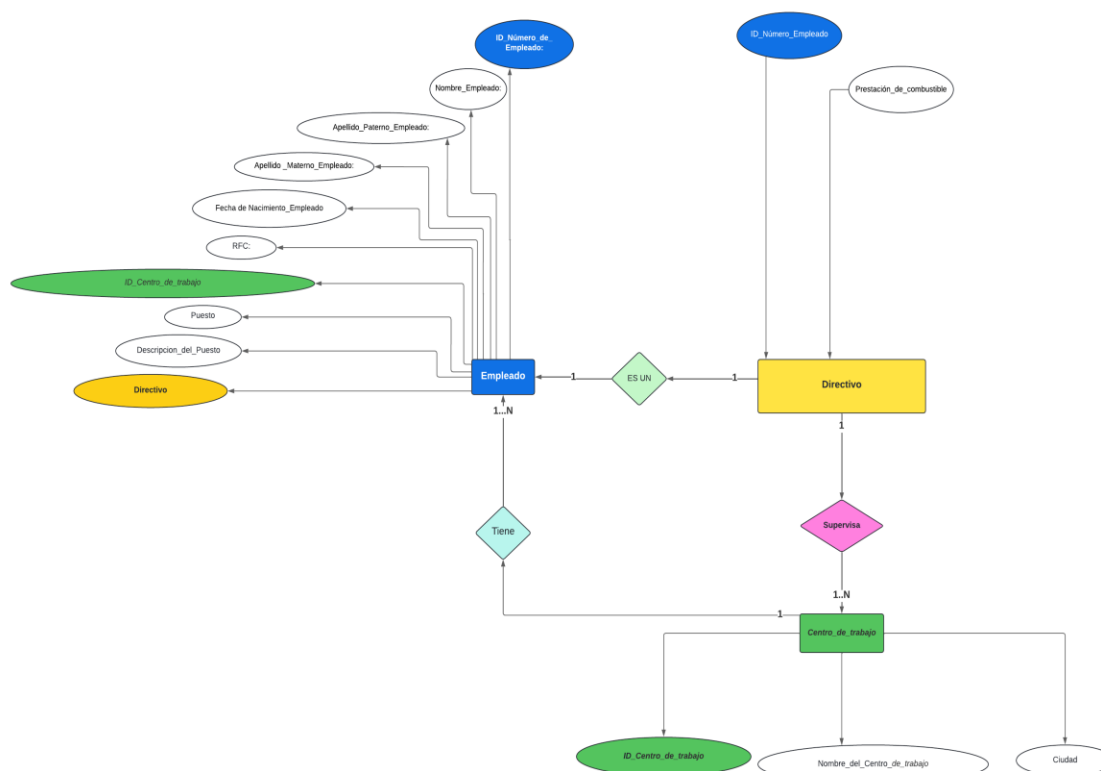
1. Analizar los datos que se presentan en la contextualización.
2. Realizar un diagrama Entidad-Relación y un diagrama Lógico-Relacional.
3. Crear la Base de Datos (BD) en Microsoft SQL Server para el sistema. El programa se desarrollará en lenguaje C++.

Nota: El desarrollo del programa en lenguaje C++ se llevará a cabo en la Actividad 2, por lo que no es necesario realizarlo en esta fase.

A continuación, se presentan en las figuras correspondientes del Diagrama Entidad-Relación (Modelo Entidad-Relación) y el Diagrama Lógico-Relacional (Modelo Logico-Racional) que se utilizarán en esta fase del proyecto:

**Figura 3**

***Modelo Entidad-Relación:***



*Nota:* El diagrama presentado es un Diagrama Entidad-Relación (ER) que modela la estructura y relaciones de una Base de Datos para una organización, específicamente enfocada en la gestión de empleados y directivos.

**Entidades:**

**Empleado (Entidad)-** en la figura de un rectángulo con esquinas redondas

**Atributos: -** en la figura de óvalos

ID\_Número\_de\_Empleado

Nombre\_Empleado

Apellido\_Paterno\_Empleado

Apellido\_Materno\_Empleado

Fecha\_de\_Nacimiento\_Empleado

RFC

Puesto

Descripción\_del\_Puesto

**Directivo (Entidad)**- en la figura de un rectángulo con esquinas redondas

**Atributos:** en la figura de óvalos

ID\_Número\_Empleado

Prestación\_de\_combustible

**Centro\_de\_trabajo (Entidad)** - en la figura de un rectángulo con esquinas redondas

**Atributos** - en la figura de óvalos:

ID\_Centro\_de\_trabajo

Nombre\_del\_Centro\_de\_trabajo

Ciudad

Relaciones -en la figura de distintos rombos:

**ES UN-** en la figura de un rombo

Relación entre "Empleado" y "Directivo"

Cardinalidad: 1 a 1

Indica que un empleado puede ser un directivo.

**Supervisa-** en la figura de un rombo

Relación entre "Directivo" y "Centro\_de\_trabajo"

Cardinalidad: 1 a N (Un directivo supervisa a múltiples centros de trabajo).

**Tiene** - en la figura de un rombo

Relación entre "Empleado" y "Centro\_de\_trabajo"

Cardinalidad: 1 a N (Un empleado está asignado a un centro de trabajo).

### **Explicación de los atributos detallados en cada entidad:**

#### **Empleado**

Cada empleado tiene un número de identificación único (ID\_Número\_de\_Empleado).

Se capturan datos personales como nombre, apellidos, fecha de nacimiento y RFC.

Se registran detalles de su puesto y la descripción del mismo.

Un empleado puede ser directivo, en cuyo caso se gestiona con la relación "ES UN".

#### **Directivo**

Identificado también por el número de empleado (ID\_Número\_Empleado), indicando que es una extensión de la entidad "Empleado".

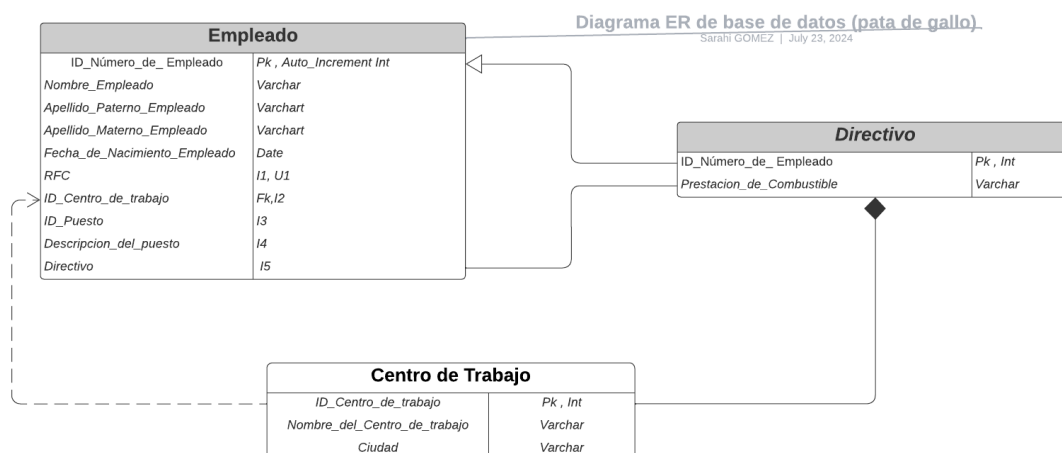
Tienen una prestación de combustible.

#### **Centro\_de\_trabajo**

Cada centro de trabajo tiene un identificador único (ID\_Centro\_de\_trabajo).

Se registran el nombre del centro y la ciudad donde se ubica.

En resumen este diagrama Entidad- Relación muestra la estructura de cómo se almacenan y relacionan los datos de empleados y directivos en una organización, así como la asignación de estos a diferentes centros de trabajo.

**Figura 4****Modelo Lógico Relacional:**

*Nota:* El segundo diagrama presentado es un Diagrama Entidad-Relación (ER) con detalles más técnicos, indicando los tipos de datos, claves primarias (PK), claves foráneas (FK), y restricciones de unicidad (U) e integridad (I). A continuación se describe la estructura del diagrama:

**1. Entidad: Empleado****Atributos:**

ID\_Número\_de\_Empleado: **PK**, Auto\_Increment Int (Clave primaria, entero con auto incremento)

Nombre\_Empleado: Varchar

Apellido\_Paterno\_Empleado: Varchar

Apellido\_Materno\_Empleado: Varchar

Fecha\_de\_Nacimiento\_Empleado: Date

RFC: **I1, U1** (Restricción de integridad y unicidad)

ID\_Centro\_de\_trabajo: **FK, I2** (Clave foránea, restricción de integridad)

ID\_Puesto: **I3** (Restricción de integridad)

Descripcion\_del\_puesto: **I4** (Restricción de integridad)

Directivo: **I5** (Restricción de integridad)

## **2. Entidad: Directivo**

### **Atributos:**

ID\_Número\_de\_Empleado: **PK**, Int (Clave primaria, entero)

Prestacion\_de\_Combustible: Varchar

## **3. Entidad: Centro de Trabajo**

### **Atributos:**

ID\_Centro\_de\_trabajo: **PK**, Int (Clave primaria, entero)

Nombre\_del\_Centro\_de\_trabajo: Varchar

Ciudad: Varchar

### **Relaciones:**

#### **1. Empleado - Directivo**

Relación entre la tabla "Empleado" y la tabla "Directivo".

**Cardinalidad:** 1 a 1 (Cada empleado puede ser directivo).

### **Claves:**

Empleado: ID\_Número\_de\_Empleado (**PK**)

Directivo: ID\_Número\_de\_Empleado (**PK**)

### **Relación de tipo Herencia:**

La entidad Directivo hereda del Empleado, lo que se refleja en que ambos comparten la clave primaria ID\_Número\_de\_Empleado, esto indica que todo **Directivo** es también un

**Empleado**, lo que implica una relación de herencia entre ambas entidades.

## **2. Empleado - Centro\_de\_Trabajo**

Relación entre la tabla "Empleado" y la tabla "Centro\_de\_Trabajo".

**Cardinalidad:** 1 a N (Un empleado está asignado a un centro de trabajo).

### **Claves:**

Empleado: ID\_Centro\_de\_trabajo (**FK, I2**)

Centro\_de\_Trabajo: ID\_Centro\_de\_trabajo (**PK**)

### **Relación de tipo Dependencia:**

La entidad Centro\_de\_Trabajo es referenciada por la entidad Empleado a través de la clave foránea ID\_Centro\_de\_trabajo, esto indica que un Empleado depende de un Centro\_de\_Trabajo para especificar su ubicación laboral.

### **Claves y Restricciones:**

**Clave Primaria (PK):** Identifica de manera única cada registro en la tabla.

Empleado: ID\_Número\_de\_Empleado

Directivo: ID\_Número\_de\_Empleado

Centro\_de\_Trabajo: ID\_Centro\_de\_trabajo

**Clave Foránea (FK):** Establece la relación entre tablas, asegurando la integridad referencial.

Empleado: ID\_Centro\_de\_trabajo (referencia a la tabla Centro\_de\_Trabajo)

**Restricciones de Integridad (I):** Aseguran que los datos cumplan con ciertas condiciones.

Empleado: RFC (**I1**), ID\_Centro\_de\_trabajo (**I2**), ID\_Puesto (**I3**),  
Descripcion\_del\_puesto (**I4**), Directivo (**I5**)



**Restricciones de Unicidad (U):** Garantizan que los valores en una columna sean únicos.

Empleado: RFC (U1)

**Descripción General:** Este diagrama proporciona una vista técnica detallada de la base de datos, especificando los tipos de datos y las restricciones aplicadas a cada campo, se observa una relación de herencia entre "Empleado" y "Directivo", donde un directivo es un tipo específico de empleado. La relación 1 a 1 entre "Empleado" y "Directivo" indica que, aunque no todos los empleados son directivos, cada directivo es necesariamente un empleado.

Por otro lado, la relación 1 a N entre "Empleado" y "Centro\_de\_Trabajo" establece que un empleado está asociado a un único centro de trabajo, mientras que un centro de trabajo puede tener múltiples empleados, esta relación de dependencia garantiza que cada empleado esté vinculado a un centro de trabajo específico.

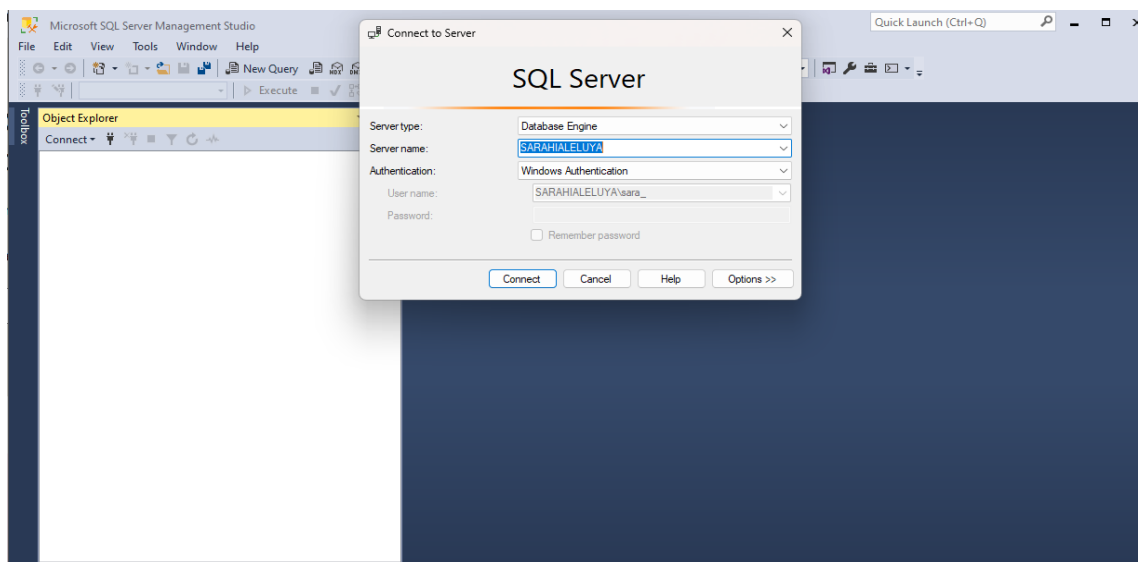
### Base de Datos (BD) en Microsoft SQL Server para el sistema:

A continuación se presentarán las figuras que muestran, paso a paso, la creación y diseño de la Base de Datos que se utilizará en SQL Server, esta Base de Datos se empleará durante las siguientes tres fases; sin embargo, en este proyecto solo se mostrará la primera fase, también se realizarán diversas consultas para verificar que los datos se hayan insertado correctamente en las tablas, además, se ejecutará una consulta que indicará si un empleado es director o no , junto con el resto de los atributos tomados de las distintas tablas de esta base de datos, las cuales son:

- **Centro\_de\_trabajo**
- **Directivo**
- **EMPLEADO**

**Figura 5**

*Gestor de Bases de Datos SQL Server:*



*Nota:* Esta imagen muestra el gestor de Bases de Datos "SQL Server", donde se creará la Base de Datos descrita en este documento, y también confirma que ya tenemos acceso a este programa en nuestro equipo.

Figura 6

**Paso 1: Creación de la Base de Datos y la tabla Centro\_de\_Trabajo (Base de datos, parte 1):**

```

ACT1_Director_Empl...LELUYA\sara_(51))  ▢ ×
-- Eliminar la base de datos
DROP DATABASE IF EXISTS ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO;
GO

-- Crear la base de datos
CREATE DATABASE ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO;
GO

-- Usar la base de datos
USE ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO;
GO

-- Crear la tabla Centro_de_Trabajo
CREATE TABLE Centro_de_Trabajo (
    ID_Centro_de_trabajo INT PRIMARY KEY,
    Nombre_del_Centro_de_trabajo VARCHAR(100) NOT NULL,
    Ciudad VARCHAR(100) NOT NULL
);
GO

```

Figura 7

**Paso 2: Creación de la tabla Empleado (Base de datos, parte 2):**

```

SQLQuery2.sql - SA...LELUYA\sara_(51))*  ▢ ×
-- Crear la tabla Centro_de_Trabajo
CREATE TABLE Centro_de_Trabajo (
    ID_Centro_de_trabajo INT PRIMARY KEY,
    Nombre_del_Centro_de_trabajo VARCHAR(100) NOT NULL,
    Ciudad VARCHAR(100) NOT NULL
);
GO

-- Crear la tabla EMPLEADO
CREATE TABLE EMPLEADO (
    ID_NUMERO_DE_EMPLEADO INT PRIMARY KEY IDENTITY (1,1),
    NOMBRE_EMPLEADO VARCHAR(50) NOT NULL,
    APELLIDO_PATERNO_EMPLEADO VARCHAR(50) NOT NULL,
    APELLIDO_MATERNO_EMPLEADO VARCHAR(50) NOT NULL,
    FECHA_NACIMIENTO_EMPLEADO DATE NOT NULL,
    RFC AS (UPPER(SUBSTRING(NOMBRE_EMPLEADO,1,2) + SUBSTRING(APELLIDO_PATERNO_EMPLEADO,1,2) +
SUBSTRING(APELLIDO_MATERNO_EMPLEADO,1,1) +
FORMAT(FECHA_NACIMIENTO_EMPLEADO,'yyMMdd'))),
    ID_CENTRO_DE_TRABAJO INT NOT NULL,
    ID_PUESTO VARCHAR(50) NOT NULL,
    DESCRIPCION_DEL_PUESTO VARCHAR(100) NOT NULL,
    DIRECTIVO BIT NOT NULL,
    FOREIGN KEY (ID_CENTRO_DE_TRABAJO) REFERENCES Centro_de_Trabajo(ID_Centro_de_trabajo)
);
GO

```

Figura 8

*Paso 3: Creación de la tabla Directivo (Base de datos, parte 3):*

```
-- Crear la tabla Directivo
CREATE TABLE Directivo (
    ID_Numero_de_Empleado INT PRIMARY KEY,
    Prestacion_de_Combustible BIT NOT NULL,
    FOREIGN KEY (ID_Numero_de_Empleado) REFERENCES EMPLEADO(ID_NUMERO_DE_EMPLEADO)
);
GO
```

00 %

Messages

Commands completed successfully.

Completion time: 2024-07-25T07:36:32.6042283-06:00

Figura 9

*Confirmación de la creación de la tabla Centro\_de\_Trabajo:*

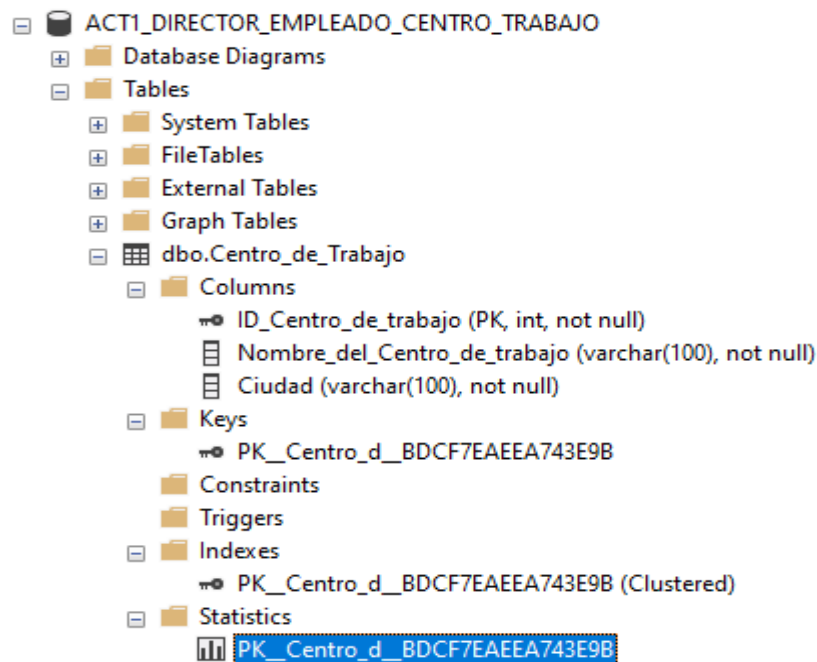


Figura 10

*Confirmación de la creación de la tabla Directivo:*

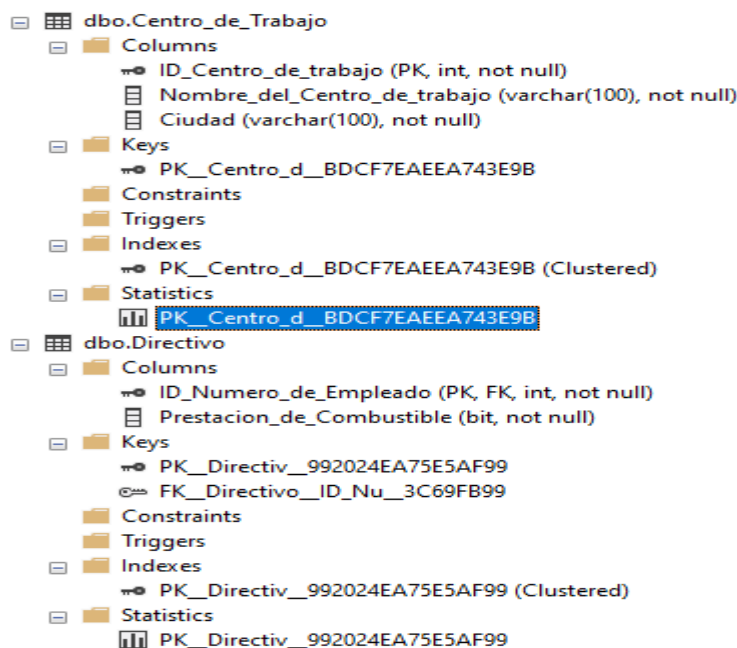


Figura 11

*Confirmación de la creación de la tabla EMPLEADO:*

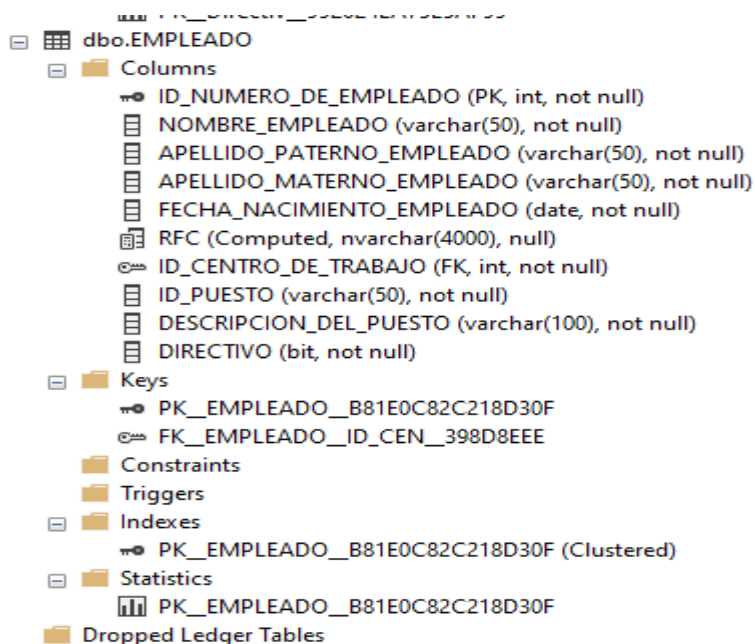
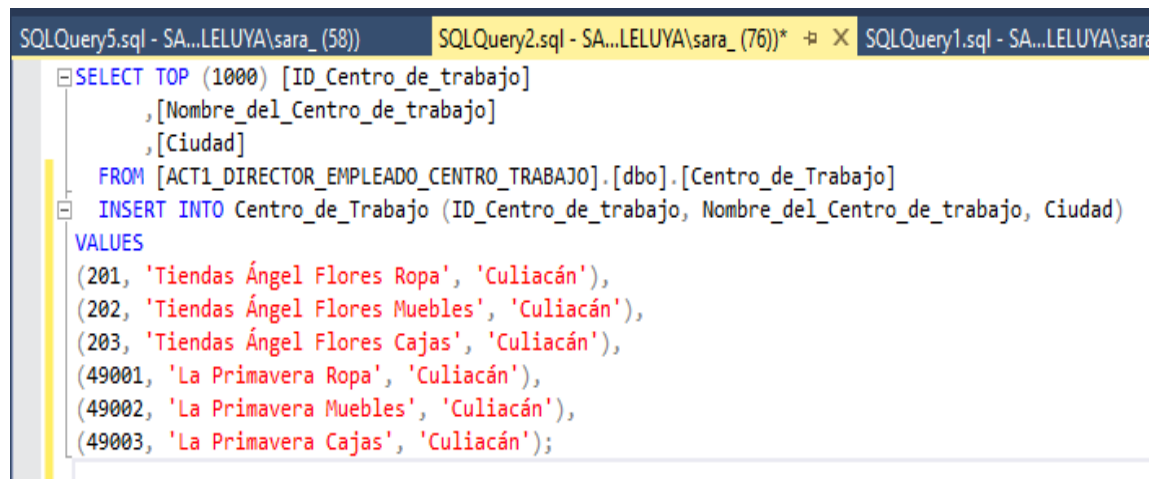


Figura 12

*Insertar datos en la tabla Centro\_de\_Trabajo:*



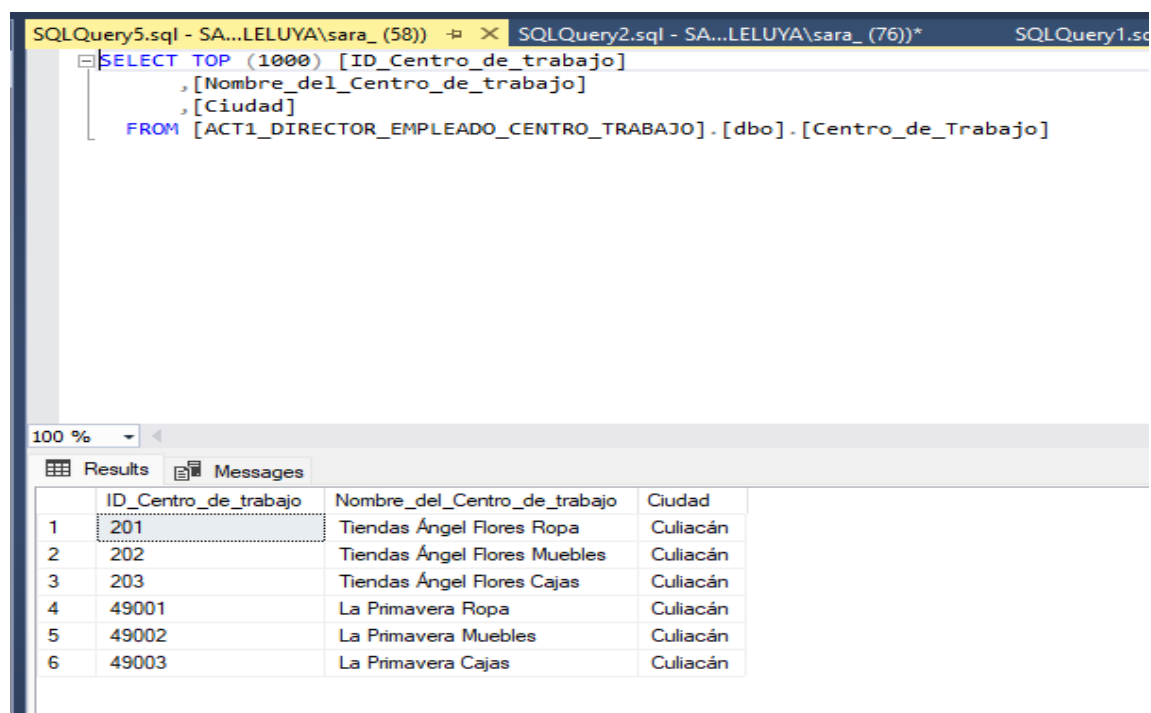
```

SQLQuery5.sql - SA...LELUYA\sara_ (58))  SQLQuery2.sql - SA...LELUYA\sara_ (76))*  SQLQuery1.sql - SA...LELUYA\sara_ (76))
SELECT TOP (1000) [ID_Centro_de_trabajo]
, [Nombre_del_Centro_de_trabajo]
, [Ciudad]
FROM [ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO].[dbo].[Centro_de_Trabajo]
INSERT INTO Centro_de_Trabajo (ID_Centro_de_trabajo, Nombre_del_Centro_de_trabajo, Ciudad)
VALUES
(201, 'Tiendas Ángel Flores Ropa', 'Culiacán'),
(202, 'Tiendas Ángel Flores Muebles', 'Culiacán'),
(203, 'Tiendas Ángel Flores Cajas', 'Culiacán'),
(49001, 'La Primavera Ropa', 'Culiacán'),
(49002, 'La Primavera Muebles', 'Culiacán'),
(49003, 'La Primavera Cajas', 'Culiacán');
  
```

Figura 13

*Consulta que confirma que se han insertado correctamente los datos en la tabla*

*Centro\_de\_trabajo:*



```

SQLQuery5.sql - SA...LELUYA\sara_ (58))  SQLQuery2.sql - SA...LELUYA\sara_ (76))*  SQLQuery1.sql - SA...LELUYA\sara_ (76))
SELECT TOP (1000) [ID_Centro_de_trabajo]
, [Nombre_del_Centro_de_trabajo]
, [Ciudad]
FROM [ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO].[dbo].[Centro_de_Trabajo]
  
```

	ID_Centro_de_trabajo	Nombre_del_Centro_de_trabajo	Ciudad
1	201	Tiendas Ángel Flores Ropa	Culiacán
2	202	Tiendas Ángel Flores Muebles	Culiacán
3	203	Tiendas Ángel Flores Cajas	Culiacán
4	49001	La Primavera Ropa	Culiacán
5	49002	La Primavera Muebles	Culiacán
6	49003	La Primavera Cajas	Culiacán

Figura 14

*Insertar datos en la tabla EMPLEADO parte 1:*

```

INSERT INTO [dbo].[EMPLEADO]
(
    [NOMBRE_EMPLEADO]
    , [APELLIDO_PATERNO_EMPLEADO]
    , [APELLIDO_MATERNO_EMPLEADO]
    , [FECHA_NACIMIENTO_EMPLEADO]
    , [ID_CENTRO_DE_TRABAJO]
    , [ID_PUESTO]
    , [DESCRIPCION_DEL_PUESTO]
    , [DIRECTIVO]
)
VALUES
(
    (<NOMBRE_EMPLEADO, varchar(50)>,
    <APELLIDO_PATERNO_EMPLEADO, varchar(50)>,
    <APELLIDO_MATERNO_EMPLEADO, varchar(50)>,
    <FECHA_NACIMIENTO_EMPLEADO, date>,
    <ID_CENTRO_DE_TRABAJO, int>,
    <ID_PUESTO, varchar(50)>,
    <DESCRIPCION_DEL_PUESTO, varchar(100)>,
    <DIRECTIVO, bit>);

GO

INSERT INTO EMPLEADO (NOMBRE_EMPLEADO, APELLIDO_PATERNO_EMPLEADO, APELLIDO_MATERNO_EMPLEADO, FECHA_NACIMIENTO_EMPLEADO, ID_CENTRO_DE_TRABAJO, ID_PUESTO, DESCRIPCION_DEL_PUESTO,
VALUES
('Jesus', 'Vega', 'Castro', '1988-03-26', 201, 'Vendedor', 'Vendedor', 0),
('Jose', 'Lopez', 'Perez', '1980-01-01', 49003, 'Gerente', 'Gerente', 1);

```

Msg 102, Level 15, State 1, Line 14  
Incorrect syntax near '<'.  
(2 rows affected)

Figura 15

*Insertar datos en la tabla EMPLEADO parte 2:*

```

[EMPLEADO]
[NOMBRE_EMPLEADO]
[APELLIDO_PATERNO_EMPLEADO]
[APELLIDO_MATERNO_EMPLEADO]
[FECHA_NACIMIENTO_EMPLEADO]
[ID_CENTRO_DE_TRABAJO]
[ID_PUESTO]
[DESCRIPCION_DEL_PUESTO]
[DIRECTIVO]

[NOMBRE_EMPLEADO, varchar(50)>,
[APELLIDO_PATERNO_EMPLEADO, varchar(50)>,
[APELLIDO_MATERNO_EMPLEADO, varchar(50)>,
[FECHA_NACIMIENTO_EMPLEADO, date>,
[ID_CENTRO_DE_TRABAJO, int>,
[ID_PUESTO, varchar(50)>,
[DESCRIPCION_DEL_PUESTO, varchar(100)>,
[DIRECTIVO, bit>);

GO

INSERT INTO EMPLEADO (NOMBRE_EMPLEADO, APELLIDO_PATERNO_EMPLEADO, APELLIDO_MATERNO_EMPLEADO, FECHA_NACIMIENTO_EMPLEADO, ID_CENTRO_DE_TRABAJO, ID_PUESTO, DESCRIPCION_DEL_PUESTO, DIRECTIVO)
VALUES
('Jesus', 'Vega', 'Castro', '1988-03-26', 201, 'Vendedor', 'Vendedor', 0),
('Jose', 'Lopez', 'Perez', '1980-01-01', 49003, 'Gerente', 'Gerente', 1);

```

Msg 102, Level 15, State 1, Line 14  
Incorrect syntax near '<'.  
(2 rows affected)

**Figura 16**

*Consulta que confirma que se han insertado correctamente los datos en la tabla EMPLEADO  
parte 1:*

SQLQuery7.sql - SA...LELUYA\sara\_ (54) | SQLQuery6.sql - SA...LELUYA\sara\_ (53)\* | SQLQuery5.sql - SA...LELUYA\sara\_ (58) | SQLQuery2.sql - SA...LELUYA\sara\_ (76)\*

```

SELECT TOP (1000) [ID_NUMERO_DE_EMPLEADO]
, [NOMBRE_EMPLEADO]
, [APELLIDO_PATERNO_EMPLEADO]
, [APELLIDO_MATERNO_EMPLEADO]
, [FECHA_NACIMIENTO_EMPLEADO]
, [RFC]
, [ID_CENTRO_DE_TRABAJO]
, [ID_PUESTO]
, [DESCRIPCION_DEL_PUESTO]
, [DIRECTIVO]
FROM [ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO].[dbo].[EMPLEADO]

```

ID_NUMERO_DE_EMPLEADO	NOMBRE_EMPLEADO	APELLIDO_PATERNO_EMPLEADO	APELLIDO_MATERNO_EMPLEADO	FECHA_NACIMIENTO_EMPLEADO	RFC	ID_CENTRO_DE_TRABAJO	ID_PUESTO	DESCRIPCION_DEL_PUESTO	DIRECTIVO
1	Jesus	Vega	Castro	1988-03-26	JEVEC880326	201	Vendedor	Vendedor	0
2	Jose	Lopez	Perez	1980-01-01	JOLOP800101	49003	Gerente	Gerente	1

**Figura 17**

*Consulta que confirma que se han insertado correctamente los datos en la tabla EMPLEADO  
parte 2:*

SQLQuery7.sql - SA...LELUYA\sara\_ (54) | SQLQuery6.sql - SA...LELUYA\sara\_ (53)\* | SQLQuery5.sql - SA...LELUYA\sara\_ (58) | SQLQuery2.sql - SA...LELUYA\sara\_ (76)\*

```

SELECT TOP (1000) [ID_NUMERO_DE_EMPLEADO]
, [NOMBRE_EMPLEADO]
, [APELLIDO_PATERNO_EMPLEADO]
, [APELLIDO_MATERNO_EMPLEADO]
, [FECHA_NACIMIENTO_EMPLEADO]
, [RFC]
, [ID_CENTRO_DE_TRABAJO]
, [ID_PUESTO]
, [DESCRIPCION_DEL_PUESTO]
, [DIRECTIVO]
FROM [ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO].[dbo].[EMPLEADO]

```

ID_NUMERO_DE_EMPLEADO	NOMBRE_EMPLEADO	APELLIDO_PATERNO_EMPLEADO	APELLIDO_MATERNO_EMPLEADO	FECHA_NACIMIENTO_EMPLEADO	RFC	ID_CENTRO_DE_TRABAJO	ID_PUESTO	DESCRIPCION_DEL_PUESTO	DIRECTIVO
1	Jesus	Vega	Castro	1988-03-26	JEVEC880326	201	Vendedor	Vendedor	0
2	Jose	Lopez	Perez	1980-01-01	JOLOP800101	49003	Gerente	Gerente	1



Figura 18

*Insertar datos en la tabla Directivo:*

```

USE [ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO]
GO

INSERT INTO [dbo].[Directivo]
(
    [ID_Numero_de_Empleado]
    , [Prestacion_de_Combustible]
)
VALUES
(
    <ID_Numero_de_Empleado, int>
    , <Prestacion_de_Combustible, bit>
)
GO

INSERT INTO Directivo (ID_Numero_de_Empleado, Prestacion_de_Combustible)
VALUES
(
    (SELECT ID_NUMERO_DE_EMPLEADO FROM EMPLEADO WHERE NOMBRE_EMPLEADO = 'Jose' AND APELLIDO_PATERNO_EMPLEADO = 'Lopez' AND APELLIDO_MATERNO_EMPLEADO = 'Perez'), 1);
  
```

100 %

Messages

Msg 102, Level 16, State 1, Line 8  
Incorrect syntax near '<'.

(1 row affected)

Completion time: 2024-07-30T21:50:38.5317177-06:00

Figura 19

*Consulta que confirma que se han insertado correctamente los datos en la tabla Directivo:*

```

SELECT TOP (1000) [ID_Numero_de_Empleado]
, [Prestacion_de_Combustible]
FROM [ACT1_DIRECTOR_EMPLEADO_CENTRO_TRABAJO].[dbo].[Directivo]
  
```

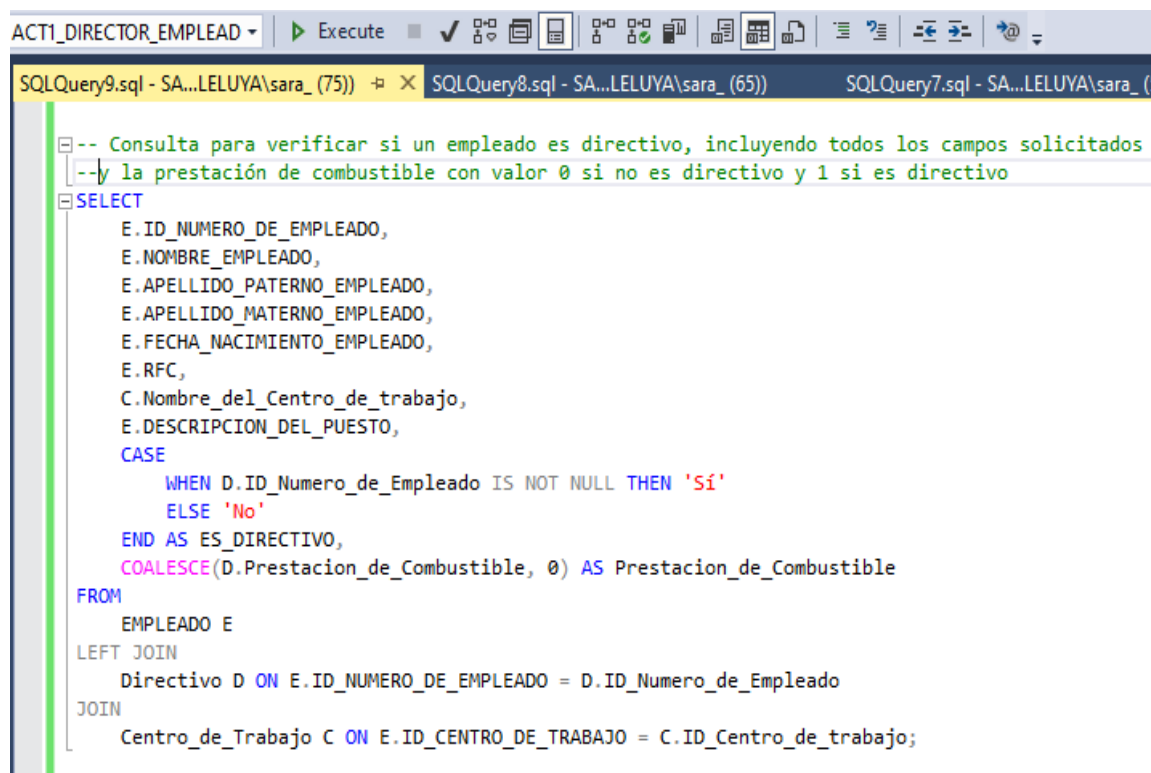
100 %

Results Messages

	ID_Numero_de_Empleado	Prestacion_de_Combustible
1	2	1

Figura 20

*Consulta para verificar si es o no Directivo el empleado (parte 1):*



```

-- Consulta para verificar si un empleado es directivo, incluyendo todos los campos solicitados
-- y la prestación de combustible con valor 0 si no es directivo y 1 si es directivo
SELECT
    E.ID_NUMERO_DE_EMPLEADO,
    E.NOMBRE_EMPLEADO,
    E.APELLIDO_PATERNO_EMPLEADO,
    E.APELLIDO_MATERNO_EMPLEADO,
    E.FECHA_NACIMIENTO_EMPLEADO,
    E.RFC,
    C.Nombre_del_Centro_de_trabajo,
    E.DESCRIPCION_DEL_PUESTO,
    CASE
        WHEN D.ID_Numero_de_Empleado IS NOT NULL THEN 'Sí'
        ELSE 'No'
    END AS ES_DIRECTIVO,
    COALESCE(D.Prestacion_de_Combustible, 0) AS Prestacion_de_Combustible
FROM
    EMPLEADO E
LEFT JOIN
    Directivo D ON E.ID_NUMERO_DE_EMPLEADO = D.ID_Numero_de_Empleado
JOIN
    Centro_de_Trabajo C ON E.ID_CENTRO_DE_TRABAJO = C.ID_Centro_de_trabajo;

```

*Nota:* La imagen muestra la consulta que está diseñada para verificar si un empleado es directivo e incluye todos los campos solicitados, también asigna un valor de prestación de combustible basado en si el empleado es directivo o no.

### Detalles de la consulta:

#### Selección de Campos:

E.ID\_NUMERO\_DE\_EMPLEADO: Número de identificación del empleado.

E.NOMBRE\_EMPLEADO: Nombre del empleado.

E.APELLIDO\_PATERNO\_EMPLEADO: Apellido paterno del empleado.

E.APELLIDO\_MATERNO\_EMPLEADO: Apellido materno del empleado.

E.FECHA\_NACIMIENTO\_EMPLEADO: Fecha de nacimiento del empleado.

E.RFC: Registro Federal de Contribuyentes del empleado.

C.Nombre\_del\_Centro\_de\_trabajo: Nombre del centro de trabajo asociado al empleado.

E.DESCRIPCION\_DEL\_PUESTO: Descripción del puesto del empleado.

ES\_DIRECTIVO: Indica si el empleado es directivo ("Sí" o "No").

Prestacion\_de\_Combustible: Valor de la prestación de combustible (1 si es directivo, 0 si no lo es).

### **Uniones (JOINS):**

**LEFT JOIN** con la tabla Directivo D: Esto asegura que todos los empleados sean incluidos en el resultado, y si un empleado es directivo (es decir, su número de empleado existe en la tabla Directivo), se incluirán los datos adicionales de directivo.

**JOIN** con la tabla Centro\_de\_Trabajo C: Esto asocia cada empleado con su centro de trabajo correspondiente.

### **Explicación de la lógica de la Prestación de Combustible:**

El campo ES\_DIRECTIVO se determina usando una condición CASE que verifica si el número de empleado existe en la tabla Directivo. Si existe, se devuelve "Sí", de lo contrario, "No".

**COALESCE** se usa para establecer el valor de la prestación de combustible, si el empleado es directivo, se toma el valor de D.Prestacion\_de\_Combustible, y si no lo es, se asigna un valor de 0.

**Figura 21**

*Consulta para verificar si es o no Directivo el empleado (parte 2):*

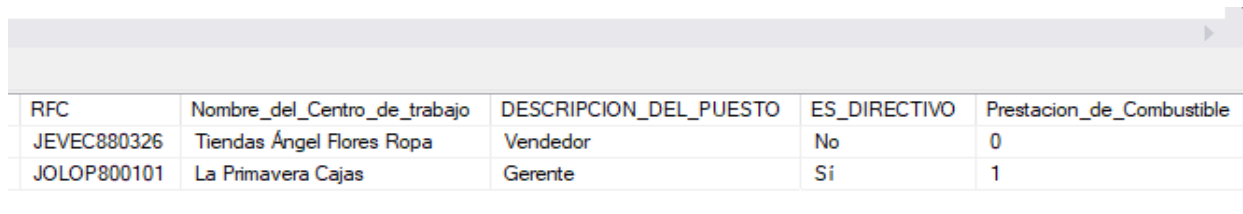


The screenshot shows a SQL query results window with a zoom level of 100%. It contains two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with 7 columns: ID\_NUMERO\_DE\_EMPLEADO, NOMBRE\_EMPLEADO, APELLIDO\_PATERNO\_EMPLEADO, APELLIDO\_MATERNO\_EMPLEADO, FECHA\_NACIMIENTO\_EMPLEADO, RFC, and Nombre\_del\_Centro\_de\_trabajo. There are two rows of data.

	ID_NUMERO_DE_EMPLEADO	NOMBRE_EMPLEADO	APELLIDO_PATERNO_EMPLEADO	APELLIDO_MATERNO_EMPLEADO	FECHA_NACIMIENTO_EMPLEADO	RFC	Nombre_del_Centro_de_trabajo
1	1	Jesus	Vega	Castro	1988-03-26	JEVEC880326	Tiendas Ángel Flores Ropa
2	2	Jose	Lopez	Perez	1980-01-01	JOLOP800101	La Primavera Cajas

**Figura 22**

*Consulta para verificar si es o no Directivo el empleado (parte 3):*



The screenshot shows a SQL query results window displaying a table with 5 columns: RFC, Nombre\_del\_Centro\_de\_trabajo, DESCRIPCION\_DEL\_PUESTO, ES\_DIRECTIVO, and Prestacion\_de\_Combustible. There are two rows of data.

RFC	Nombre_del_Centro_de_trabajo	DESCRIPCION_DEL_PUESTO	ES_DIRECTIVO	Prestacion_de_Combustible
JEVEC880326	Tiendas Ángel Flores Ropa	Vendedor	No	0
JOLOP800101	La Primavera Cajas	Gerente	Sí	1

### **Conclusión:**

La importancia de adquirir conocimientos que aplican los conceptos básicos de la Programación Orientada a Objetos (POO) es fundamental tanto en la vida cotidiana como en el ámbito laboral por varias razones, las cuales son la fomentación de una forma de pensar estructurada y lógica, lo que puede ayudar a resolver problemas cotidianos de manera más eficiente, mediante programas que automaticen tareas repetitivas, ahorrando tiempo y esfuerzo, así mismo refuerza el pensamiento crítico y lógico, habilidades útiles en cualquier aspecto de la vida, el entender cómo funcionan las aplicaciones y sistemas que usamos a diario actualmente es una ventaja.

Muchas empresas buscan profesionales con habilidades en POO debido a su aplicabilidad en el desarrollo de software moderno, la POO permite la creación de módulos independientes (clases y objetos), facilitando el trabajo en equipo y la colaboración en proyectos grandes, ya que promueve la reutilización de código a través de la herencia y los objetos, haciendo que los proyectos sean más eficientes y fáciles de mantener, es más sencillo desarrollar software que pueda escalar y adaptarse a nuevas funcionalidades sin necesidad de reescribir grandes partes del código, ayudando a modelar sistemas complejos de manera más intuitiva y organizada, lo que es crucial en campos como la ingeniería de software, finanzas, salud, y más.

En resumen, el aprendizaje de la Programación Orientada a Objetos proporciona una base sólida para abordar problemas complejos de manera efectiva y abre muchas puertas en el mercado laboral, especialmente en roles relacionados con el desarrollo de software y la tecnología.

Al aplicar estos conocimientos, sobre el diseño de bases de datos en SQL Server es muy

valioso tanto en la vida cotidiana como en el ámbito laboral, por qué: ayuda a saber diseñar Bases de Datos, esta te permite organizar y gestionar tus propios datos personales, como contactos, colecciones, y registros de gastos de manera más eficiente, dentro de la vida cotidiana el crear bases de datos, automatiza tareas domésticas o personales, como el seguimiento de inventarios caseros, listas de compras, y planificación de eventos.

Las habilidades en SQL Server y diseño de Bases de Datos son altamente demandadas en el mercado laboral, especialmente en roles relacionados con la gestión de datos, desarrollo de software y análisis de datos, un buen diseño de Base de Datos permite un acceso y manipulación de datos más eficiente, mejorando la productividad y el rendimiento de las aplicaciones y sistemas, el conocimiento de SQL Server te permite implementar medidas de integridad y seguridad de datos, asegurando que la información sea precisa y esté protegida contra accesos no autorizados, en muchos sectores, las decisiones se basan en el análisis de datos, el diseño de bases de datos eficientes permite almacenar y consultar datos de manera efectiva, proporcionando la base para una toma de decisiones informada, la adaptabilidad a nuevas tecnologías, como la inteligencia artificial y el análisis de big data, requieren una base sólida en Gestión de Bases de Datos, con conocimientos en SQL Server, se está mejor preparado para adaptarse a estas tendencias, ya que estas facilitan la colaboración entre diferentes departamentos y equipos dentro de una empresa, mejorando la comunicación y el flujo de información.

En resumen, aprender a diseñar una Base de Datos en SQL Server no solo mejora la capacidad para gestionar datos de manera eficiente y segura, sino que también te hace más competitivo en el mercado laboral, abriendo oportunidades en diversas industrias que dependen de la gestión y el análisis de datos.

Pero para ellos es fundamental adquirir conocimientos sobre cómo realizar el Modelo

Lógico-Racional y el Modelo Entidad-Relacional, esto trae varios beneficios esenciales en la vida cotidiana como en el ámbito laboral, estos modelos permiten estructurar datos de manera lógica y organizada, lo que es útil para gestionar información personal como presupuestos, listas de contactos, o inventarios caseros, si se trabaja en proyectos personales que implican bases de datos, como el desarrollo de aplicaciones o blogs, estos modelos ayudarán a diseñar sistemas de datos más eficientes y coherentes, aprender a diseñar estos modelos mejora las habilidades analíticas, ayudando a comprender y resolver problemas complejos de manera más estructurada y lógica.

Los Modelos Lógico-Racional y Entidad-Relacional son fundamentales para el diseño de bases de datos, conocer estos modelos te permite crear bases de datos bien estructuradas y eficientes, un buen diseño de Base de Datos es crucial para el desarrollo de aplicaciones y sistemas de software, asegurando que sean escalables, mantenibles y eficientes, en el entorno empresarial, estos modelos ayudan a diseñar sistemas de información que optimizan la gestión de datos, mejorando la toma de decisiones y la eficiencia operativa, entender y usar estos modelos facilita la comunicación entre los diferentes equipos de desarrollo y administración, ya que proporcionan una representación clara y común de los datos y sus relaciones, en muchas industrias, el diseño adecuado de bases de datos es crucial para cumplir con normativas y estándares de gestión de datos y privacidad, ya que muchas tecnologías avanzadas, como el análisis de big data y la inteligencia artificial, requieren un sólido conocimiento del diseño de bases de datos, estos modelos proporcionan la base para implementar y utilizar estas tecnologías de manera efectiva.

El Modelo ER permite visualizar cómo se relacionan las diferentes entidades en un sistema, facilitando el diseño y la comprensión de la estructura de datos, ayuda a simplificar y

organizar el diseño de Bases de Datos, asegurando que todas las relaciones y entidades necesarias están correctamente definidas.

Por su parte el **Modelo Lógico-Racional**: se enfoca en la eficiencia de la estructura de datos y en la optimización del acceso y manipulación de datos, proporcionando una guía clara para la implementación técnica de bases de datos en sistemas de gestión de bases de datos relacionales (RDBMS).

En pocas palabras aprender a realizar el Modelo Lógico-Racional y el Modelo Entidad-Relacional es esencial para diseñar sistemas de datos eficientes y efectivos, lo que es beneficioso tanto en la gestión de información personal como en el desarrollo y mantenimiento de sistemas empresariales y tecnológicos.

Así mismo aprender las diferentes herencias en la Programación Orientada a Objetos (POO) es valioso tanto en la vida cotidiana como en el ámbito laboral, porque al entender la herencia en POO ayuda a descomponer problemas complejos en partes manejables, lo que es útil para planificar proyectos personales o solucionar problemas en casa, la herencia en POO permite la reutilización de código y conceptos, lo que es análogo a aplicar soluciones previas a nuevos problemas en la vida diaria, haciendo más eficiente el aprendizaje y la resolución de problemas, mejorando las habilidades de pensamiento lógico y estructurado, lo que es beneficioso en cualquier situación que requiera planificación y organización.

De igual modo es útil en el ámbito laboral, ya que la herencia permite crear jerarquías de clases que reutilizan y extienden funcionalidades, haciendo que el diseño de software sea más eficiente y modula los sistemas que utilizan herencia, son más fáciles de mantener y actualizar, ya que los cambios en una clase base se propagan a las clases derivadas, reduciendo la duplicación de código, facilitando la escalabilidad de proyectos de software, permitiendo añadir



nuevas funcionalidades sin reescribir grandes partes del código existente, mejorando la colaboración entre desarrolladores, ya que proporciona una estructura clara y organizada para el código, facilitando el entendimiento y la colaboración en proyectos grandes.

La herencia y otros conceptos de POO son aplicables en diversos sectores como la inteligencia artificial, el desarrollo de videojuegos, aplicaciones móviles, sistemas embebidos y más, las tecnologías emergentes a menudo se construyen sobre principios de POO, por lo que entender la herencia nos prepara para trabajar con herramientas y lenguajes modernos.

### **Tipos de Herencia:**

**Herencia Simple:** Una clase derivada hereda de una sola clase base, lo que facilita la comprensión y el uso de funcionalidades específicas.

**Herencia Múltiple:** Una clase puede heredar de múltiples clases base, combinando funcionalidades diversas, aunque esto puede añadir complejidad y requiere un manejo cuidadoso.

**Herencia Multinivel:** Una clase hereda de otra clase derivada, creando una jerarquía que puede modelar relaciones más complejas y específicas.

**Herencia Jerárquica:** Varias clases derivadas heredan de una sola clase base, lo que permite compartir una base común de funcionalidad mientras se especializan en diferentes áreas.

**Herencia Híbrida:** Combinación de los tipos anteriores, utilizada en situaciones donde se necesita una estructura más flexible y compleja.

Los beneficios de conocer diferentes Herencias son que permite elegir el tipo de herencia más adecuado según el problema a resolver, optimizando el diseño de software, facilitando la implementación de nuevas características y el mantenimiento del código existente, un buen uso de la herencia puede mejorar la robustez, reutilización y claridad del código, resultando en software de mayor calidad.

En resumen, conocer las diferentes herencias en POO proporciona herramientas poderosas para diseñar y mantener software eficiente y escalable, además de mejorar las habilidades de resolución de problemas y pensamiento lógico, útiles tanto en la vida cotidiana como en el entorno laboral.

## Referencias

Ag Collage. (s. f.). Lenguajes de Programación II Actividad 1. Creación de la base de datos. *Lenguajes de Programación II Actividad 1. Creación de la Base de Datos*. Recuperado el 02 de agosto de 2024 de:

[https://agcollege.edu.mx/literaturas/59/15/Actividad\\_1\\_Lenguajes\\_de\\_Programacion\\_II\\_V5.docx.pdf](https://agcollege.edu.mx/literaturas/59/15/Actividad_1_Lenguajes_de_Programacion_II_V5.docx.pdf)

Blanca Guillen. (2022, 12 junio). *Método de Gauss-Seidel* [Vídeo]. YouTube.

Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=F0PLTftQqJk>

*Blog IfGeekThen*. (s. f.). IfgeekthenNTTDATA. Recuperado el 02 de agosto de 2024 de:

<https://ifgeekthen.nttdata.com/s/post/herencia-en-programacion-orientada-objetos-MCPV3PCZDNBFHSROCCU3JMI7UIJQ?language=es>

campusMVP. (s. f.). *Diseñando una base de datos en el modelo relacional* - campusMVP.es. campusMVP.es. Recuperado el 02 de agosto de 2024 de:

<https://www.campusmvp.es/recursos/post/Disenando-una-base-de-datos-en-el-modelo-relacional.aspx>

Código Correcto. (2023, 12 septiembre). *Tipos de Bases de Datos (SQL y NoSQL)* ¿Cuándo y por qué usar cada una en 2023? [Vídeo]. YouTube.

Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=GDCZsu3Gfmo>

*Comprender y evaluar las relaciones entre tablas* / Adobe Commerce. (s. f.).

Recuperado el 02 de agosto de 2024 de:

<https://experienceleague.adobe.com/es/docs/commerce-business-intelligence/mbi/analyze/warehouse-manager/table-relationships>

David Josafat Santana Cobian. (2022a, septiembre 3). *01 Introducción al Análisis Numérico con ayuda de R Aproximar el área bajo una curva* [Vídeo]. YouTube.

Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=tHwVgFKvWcI>

David Josafat Santana Cobian. (2022b, septiembre 7). *02 Introducción al Análisis Numérico con ayuda de R Método de bisección 1 de 3* [Vídeo]. YouTube.

Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=6wfe3AK4sD4>

David Josafat Santana Cobian. (2022c, septiembre 10). *03 Introducción al Análisis Numérico con ayuda de R Método de bisección 2 de 3* [Vídeo]. YouTube.

Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=j2bfgRugpAA>

David Josafat Santana Cobian. (2022d, septiembre 14). *04 Introducción al Análisis Numérico con ayuda de R Método de bisección 3 de 3* [Vídeo]. YouTube.

Recuperado el 02 de agosto de 2024 de:

[https://www.youtube.com/watch?v=hsC5-uqrJ\\_g](https://www.youtube.com/watch?v=hsC5-uqrJ_g)

David Josafat Santana Cobian. (2022e, septiembre 21). *05 Introducción al Análisis Numérico con ayuda de R Newton Raphson para aproximar raíces y formatos* [Vídeo].

Recuperado el 02 de agosto de 2024 de:

YouTube. <https://www.youtube.com/watch?v=QJDM8kRBskE>

*Diagrama entidad relación: ¿Qué es para qué sirve? Con ejemplos / Miro.* (s. f.).

Recuperado el 02 de agosto de 2024 de:

<https://miro.com/>. <https://miro.com/es/diagrama/que-es-diagrama-entidad-relacion/>

*DISEÑO LÓGICO DE BASES DE DATOS RELACIONALES*. (s. f.).

EDteam. (2020, 14 febrero). *¿Qué son las BASES DE DATOS? - La mejor explicación en español* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=knVwokXITGI>

Iammalf. (2023, 11 marzo). Conceptos básicos de la programación orientada a objetos (POO). *Diseño de Paginas Web Cusco*. Recuperado el 02 de agosto de 2024 de:

<https://webdesigncusco.com/conceptos-basicos-de-la-programacion-orientada-a-objetos-poo/>

IQ. Ab. (2017, 8 marzo). *Método de Bisección usando excel, ejemplo 1* [Vídeo].

YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=-dFSp-Ge79w>

Isabel Cruz Granados. (2022, 24 junio). *Diseño lógico. Paso del modelo ER al Relacional. Veterinaria* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=QPfsZrrchJw>

Larque, J. (2024, 4 junio). *Tipos de bases de datos: cuáles hay y por qué es importante elegirlos bien*. Blog de Hiberus. Recuperado el 02 de agosto de 2024 de:

<https://www.hiberus.com/crecemos-contigo/tipos-de-bases-de-datos-cuales-hay-y-por-que-es-importante-elegirlos-bien/>

asmatematicas.es. (2011, 3 octubre). *Método de Jacobi (Universidad)* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=Q6BdR45uo7I>

León, N. (2017, 5 abril). *Herencia y tipos en la Programación Orientada a Objetos - C#*: *Programación orientada a objetos* [Vídeo]. LinkedIn.

Recuperado el 02 de agosto de 2024 de:

<https://es.linkedin.com/learning/c-sharp-programacion-orientada-a-objetos/herencia-y-tipos-en-la-programacion-orientada-a-objetos#:~:text=La%20herencia%20se%20clasifica%20en,cuales%20se%20hereda%20una%20clase>.

Les MathsMatiques. (2021, 11 octubre). *Método de Bisección usando EXCEL. Incluye plantilla*. [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=vsZ-36fs2f8>

Matemáticas con Carito. (2020, 9 octubre). *Método de bisección* [Vídeo]. YouTube.

Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=mdG6gpzE54k>

NetMentor. (s. f.-a). *Encapsulamiento en programacion orientada a objetos*.

Recuperado el 02 de agosto de 2024 de:

<https://www.netmentor.es/entrada/encapsulamiento-poo>.

<https://www.netmentor.es/entrada/encapsulamiento-poo>

NetMentor. (s. f.-b). *Herencia en programación orientada a objetos*.

Recuperado el 02 de agosto de 2024 de:

<https://www.netmentor.es/entrada/herencia-poo>.

<https://www.netmentor.es/entrada/herencia-poo#:~:text=la%20clase%20padre->

[,1%20%2D%20Herencia%20en%20POO,otras%20clases%20que%20ser%C3%A1n%20padres](https://www.netmentor.es/entrada/herencia-poo#:~:text=la%20clase%20padre-).

NetMentor. (s. f.-c). *Interfaces en programcion orientada a objetos*.

Recuperado el 02 de agosto de 2024 de:

<https://www.netmentor.es/entrada/interfaces-poo>.

<https://www.netmentor.es/entrada/interfaces-poo>

NetMentor. (s. f.-d). *Polimorfismo en programación orientada a objetos*.

Recuperado el 02 de agosto de 2024 de:

<https://www.netmentor.es/entrada/polimorfismo-poo>.

<https://www.netmentor.es/entrada/polimorfismo-poo>

Nuevo Espacio CECE. (2020, 4 abril). *Análisis Numérico. Método de Bisección en R*.

Prof. Darío Bacchini [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=7OfJSQown8U>

Por. (2022, 3 noviembre). ¿En qué consiste el método de Jacobi para resolver sistemas de ecuaciones? *TU BLOG INFORMATIVO*. Recuperado el 02 de agosto de 2024 de:

<https://tubloginformativo.com/en-que-consiste-el-metodo-de-jacobi-para-resolver-sistemas-de-ecuaciones/>

*Qué es un diagrama entidad-relación*. (s. f.-a). Lucidchart.

Recuperado el 02 de agosto de 2024 de:

<https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion>

*Qué es un diagrama entidad-relación*. (s. f.-b). Lucidchart.

Recuperado el 02 de agosto de 2024 de:

<https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion>

¿Qué es una base de datos? (s. f.). Nutanix. Recuperado el 02 de agosto de 2024 de:

<https://www.nutanix.com/mx/info/database>

¿Qué es una base de datos? - Explicación de las bases de datos en la nube - AWS. (s. f.).

Amazon Web Services, Inc. Recuperado el 02 de agosto de 2024 de:

<https://aws.amazon.com/es/what-is/database/>

Sebastián Macías. (2022, 6 abril). *Métodos numéricos, Bisección / Programación en R* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

[https://www.youtube.com/watch?v=RL\\_G9oUiTJA](https://www.youtube.com/watch?v=RL_G9oUiTJA)

*SOLUCIONES DE ECUACIONES ALGEBRAICAS*. (s. f.).

Termofluidos RMM. (2023a, octubre 9). *5 pasos del Método de la Bisección* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=HiwG2gBgl-A>

Termofluidos RMM. (2023b, octubre 9). *Ciclos en Rstudio* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=fsRWtAMR2AA>

Termofluidos RMM. (2023c, octubre 9). *Condicionales en Rstudio* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=bbwWI4g6ZUI>

Termofluidos RMM. (2023d, octubre 9). *Operaciones básicas en R Studio* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=aHnJ4KyZ1D0>

Termofluidos RMM. (2023e, octubre 13). *5 consejos: el Método de Bisección* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=0nHEA6j4Qsc>

Termofluidos RMM. (2023f, octubre 13). *Método de Bisección en Rstudio* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:



<https://www.youtube.com/watch?v=1uic7Zf7vpM>

UNAM-DGAPA-PAPIME [UNAM-DGAPA-PAPIME]. (2019). *Métodos iterativos de Jacobi y Gauss-Seidel*. Recuperado el 02 de agosto de 2024 de:

[https://www.ingenieria.unam.mx/pinilla/PE105117/pdfs/tema3/3-3\\_metodos\\_jacobi\\_gauss-seidel.pdf](https://www.ingenieria.unam.mx/pinilla/PE105117/pdfs/tema3/3-3_metodos_jacobi_gauss-seidel.pdf)

Vanessa Pedrozo. (2014, 25 septiembre). *Método de bisección - métodos numéricos* [Vídeo]. YouTube. Recuperado el 02 de agosto de 2024 de:

<https://www.youtube.com/watch?v=0WPixuL6AZU>

*Video Conferencing, Web Conferencing, Webinars, Screen Sharing*. (s. f.). [Vídeo]. Zoom. Recuperado el 02 de agosto de 2024 de:

<https://academiaglobal-mx.zoom.us/rec/share/Y46KSrCq4zXKSrK9sWgmYajxB1DjbWaVzo6LgwjufPnOCDFxV9poPsnOF7umjNhi.9ZOtI--GrCwfYYpg>

*Video conferencing, web conferencing, webinars, screen sharing*. (s. f.-a). Zoom.

Recuperado el 02 de agosto de 2024 de:

[https://academiaglobal-mx.zoom.us/rec/play/hMtmzkwijwqpk6y0C3-RoAAKKASiUWmkC6mElH6DwnT8tDvdj2ZJP3OuZeGOZ4AUoxYPxvE8fniYHrGE.KZWWNRoThvnhlCy?canPlayFromShare=true&from=share\\_recording\\_detail&continueMode=true&componentName=rec-play&originRequestUrl=https%3A%2F%2Facademiaglobal-mx.zoom.us%2Frec%2Fshare%2FIBVH5Xa7Yknv-tdSGGWNKEe9itEPMIVQLBnUWeZoBoQNpdK5YfJlGMJ-nJ3Eftb4.mjp8XyMMktXB3hf7](https://academiaglobal-mx.zoom.us/rec/play/hMtmzkwijwqpk6y0C3-RoAAKKASiUWmkC6mElH6DwnT8tDvdj2ZJP3OuZeGOZ4AUoxYPxvE8fniYHrGE.KZWWNRoThvnhlCy?canPlayFromShare=true&from=share_recording_detail&continueMode=true&componentName=rec-play&originRequestUrl=https%3A%2F%2Facademiaglobal-mx.zoom.us%2Frec%2Fshare%2FIBVH5Xa7Yknv-tdSGGWNKEe9itEPMIVQLBnUWeZoBoQNpdK5YfJlGMJ-nJ3Eftb4.mjp8XyMMktXB3hf7)

*Video conferencing, web conferencing, webinars, screen sharing*. (s. f.-b). Zoom.

Recuperado el 02 de agosto de 2024 de:

https://academiaglobal-  
mx.zoom.us/rec/play/Tzz5uLncPcYurNpIA3rmnDlsGWTxvIQjU85TT25yRP-  
YlnBYhZ7Kw26VwhYzVQ7Udybk1duiBDcaQR9h.hzDwhr3wfyQQCMZ8?canPlayFromShare  
=true&from=share\_recording\_detail&continueMode=true&componentName=rec-  
play&originRequestUrl=https%3A%2F%2Facademiaglobal-mx.zoom.us%2Frec%2Fshare%2F-  
m8m5Aj5kX-  
Vx7iDvp6kbWEOOYR9OWkwdHxv0trK8RNU15sdXPHfVKIRIV7bVg44.nkaQpdAHcQ3yz\_e  
g

*Video conferencing, web conferencing, webinars, screen sharing.* (s. f.-c). Zoom.

Recuperado el 02 de agosto de 2024 de:

https://academiaglobal-mx.zoom.us/rec/play/d2d7SDh985dwvn46OWIVpMzc-  
21TFDo2dtcPcdOAS9atYYB4ZtIJ-8\_u5gyu2ezNoQXU-  
SHxdz4MNxp3.4Ss0av7Qn4jVyhng?canPlayFromShare=true&from=share\_recording\_detail&c  
ontinueMode=true&componentName=rec-  
play&originRequestUrl=https%3A%2F%2Facademiaglobal-  
mx.zoom.us%2Frec%2Fshare%2FFCU6n0G90oCKLuF\_d-  
xusMXyIk9Ve9Zjc8vaaa0OKEfKiOsl0vM4ozDvQzXffYye.SIwJqtHUPbCjB-Q3