IES Virgen del Carmen

JUNTA DE ANDALUCIA
CONSEJERÍA DE EDUCACIÓN

Departamento de Informática

Modelo Entidad Relación, ejercicios resueltos

Juan Gualberto

Noviembre 2024

UNIÓN EUROPEA
Fondo Social Europeo
El FSE invierte en tu futuro

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Índice

La librería 2
Solución 2
Explicación 4
Compra-Venta Coches 4
Solución 5
Explicación 5
Cadena de Gimnasios 7
Solución 8
Explicación 8
Proyectos software 10
Solución 11
Explicación 13

La librería

Una librería quiere implementar un sistema para gestionar la información de sus libros, autores y ventas. La librería necesita tener la siguiente información:

1. **Libros**: Cada libro tiene un identificador único, un título, una fecha de publicación, y un precio. Cada libro puede ser escrito por uno o varios autores.
2. **Autores**: Cada autor tiene un identificador único, nombre y país de origen.
3. **Ventas**: La librería desea registrar cada venta realizada, incluyendo la fecha de venta, el cliente que realizó la compra, y el total de la venta. Un cliente puede comprar uno o varios libros en una misma venta, y cada venta tiene un identificador único.
4. **Clientes**: La librería necesita la información básica de los clientes: identificador único, nombre y correo electrónico.

Relaciones: - Un libro puede tener múltiples autores (relación “escrito_por”). - Un autor puede haber escrito múltiples libros. - Una venta puede incluir varios libros (relación “incluye”). - Un cliente puede realizar múltiples ventas.

Modelar las entidades, sus atributos, y relaciones usando un diagrama entidad-relación.

Solución

```
@startuml
entity "Libro" {
    +ID_Libro : int
    Titulo : string
    Fecha_Publicacion : date
    Precio : float
}

entity "Autor" {
    +ID_Autor : int
    Nombre : string
    Pais : string
}

entity "Cliente" {
    +ID_Cliente : int
    Nombre : string
    Correo : string
}

entity "Venta" {
```

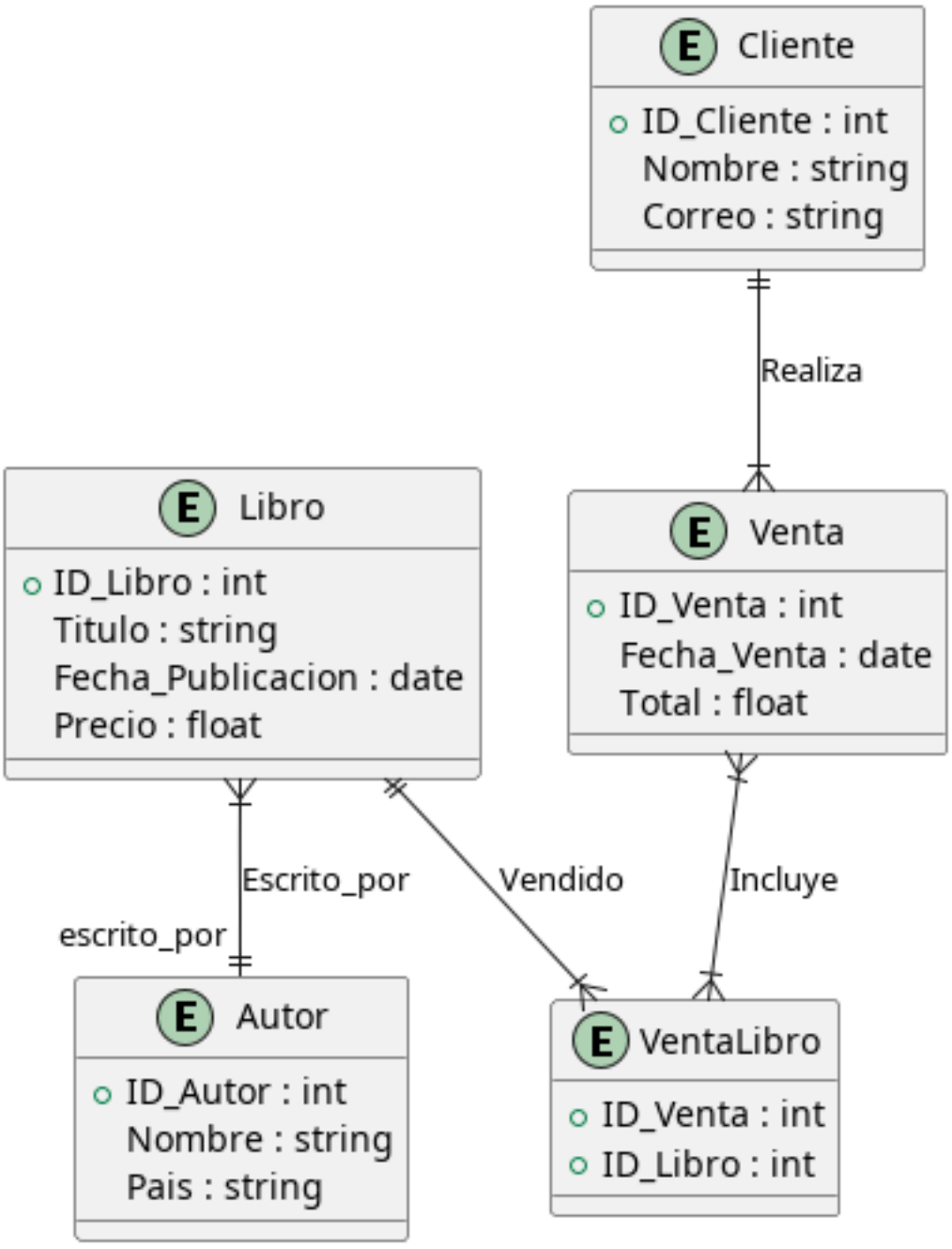

Figura 1: Solución

```
    +ID_Venta : int
    Fecha_Venta : date
    Total : float
}

entity "VentaLibro" {
    +ID_Venta : int
    +ID_Libro : int
}

' Relación entre entidades
Libro }|--|| "escrito_por" Autor : Escrito_por
Cliente ||--|{ Venta : Realiza
Venta }|--|{ VentaLibro : Incluye
Libro ||--|{ VentaLibro : Vendido

@enduml
```

Explicación

- **Entidad “Libro”** y **Entidad “Autor”** tienen una relación muchos-a-muchos (“escrito_por”) entre ellos.
- **Entidad “Venta”** y **Entidad “Cliente”** tienen una relación uno-a-muchos, ya que un cliente puede realizar varias ventas.
- **Entidad “Venta”** y **Entidad “Libro”** se relacionan a través de la entidad **“VentaLibro”** para registrar múltiples libros en una misma venta.

Compra-Venta Coches

Una agencia de vehículos quiere implementar un sistema para gestionar la compra y venta de autos. La agencia necesita registrar la siguiente información:

1. **Vehículos**: Cada vehículo tiene un identificador único, marca, modelo, año, precio y tipo de combustible.
2. **Clientes**: Los clientes que compran o venden vehículos tienen un identificador único, nombre, número de teléfono y dirección.
3. **Ventas**: Cada venta realizada tiene una fecha de venta, el monto total de la venta y el cliente que realiza la compra.
4. **Compras**: Cada compra de un vehículo a un cliente tiene una fecha de compra, precio de compra y cliente que realiza la venta.

Relaciones: - Un cliente puede vender varios vehículos a la agencia, pero un vehículo sólo puede ser

vendido una vez. - Un cliente puede comprar múltiples vehículos, pero cada venta es independiente. - Cada vehículo puede estar en una de las transacciones (compra o venta).

Solución

```
@startuml
entity "Vehículo" {
    +ID_Vehiculo : int
    Marca : string
    Modelo : string
    Año : int
    Precio : float
    Combustible : string
}

entity "Cliente" {
    +ID_Cliente : int
    Nombre : string
    Telefono : string
    Direccion : string
}

entity "Venta" {
    +ID_Venta : int
    Fecha_Venta : date
    Total : float
}

entity "Compra" {
    +ID_Compra : int
    Fecha_Compra : date
    Precio_Compra : float
}

' Relación entre entidades
Cliente ||--o{ Venta : Realiza
Cliente ||--o{ Compra : Ofrece
Venta }|--|| Vehículo : Incluye
Compra }|--|| Vehículo : Comprado

@enduml
```

Explicación

- **Entidad “Vehículo”** se conecta a **“Venta”** y **“Compra”** para diferenciar entre cuando un vehículo es comprado por la agencia o vendido a un cliente.

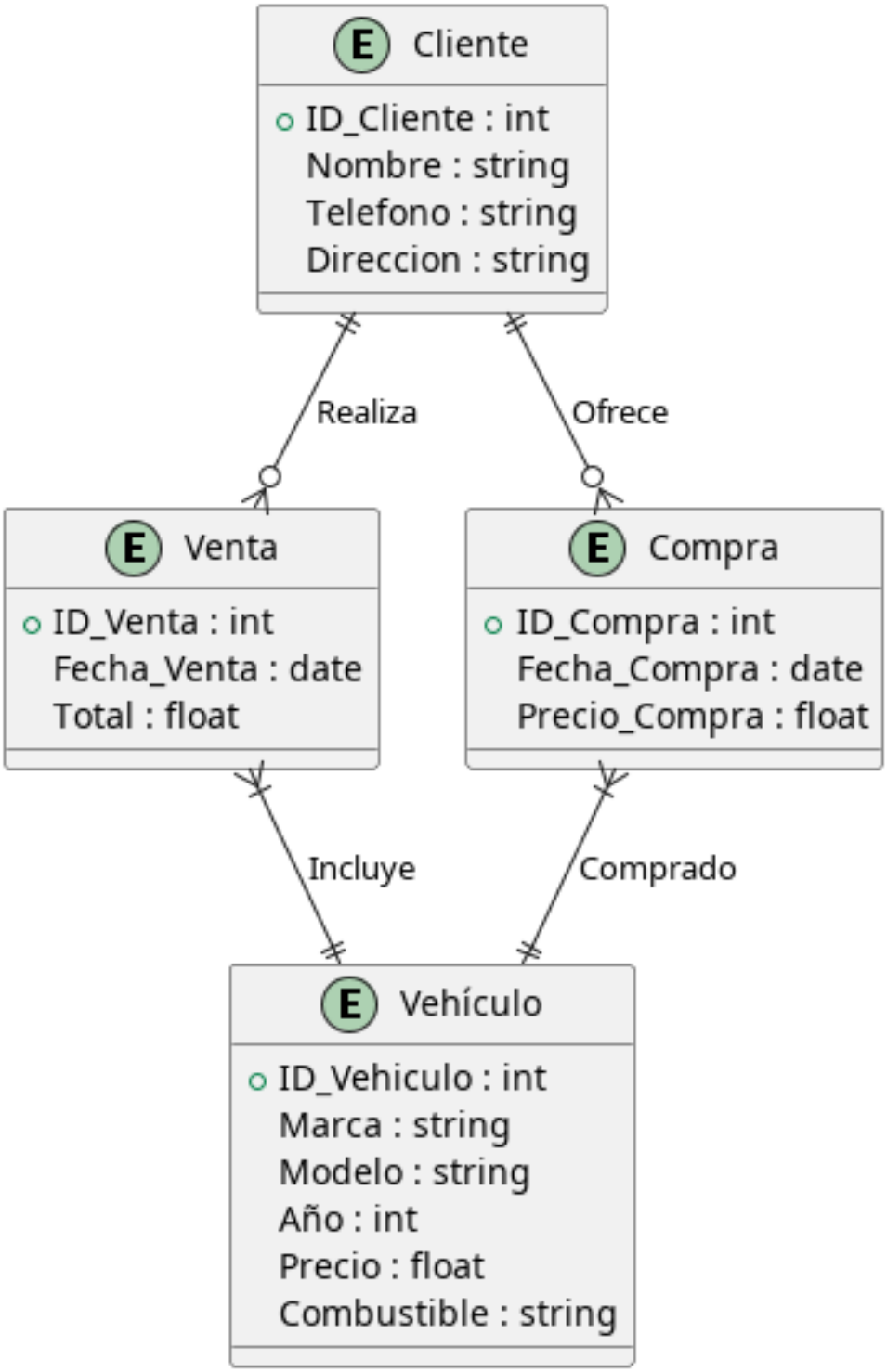

Figura 2: Solcución

- **Entidad "Cliente"** está relacionada con **"Venta"** y **"Compra"** para reflejar la transacción correspondiente (compra o venta).
- Las relaciones aseguran que un cliente pueda realizar varias ventas o compras, pero cada vehículo sólo estará en una transacción a la vez, de compra o venta.

Cadena de Gimnasios

Una cadena de gimnasios desea organizar la información de sus **Clientes**, **Entrenadores**, **Planes de Membresía** y las **Sesiones** que los clientes reservan. Las especificaciones son las siguientes:

1. **Cliente**:
    - Cada cliente tiene un **ID único**.
    - Se almacenan su **nombre**, **correo electrónico**, **fecha de registro** y **número de teléfono**.
2. **Entrenador**:
    - Cada entrenador tiene un **ID único**.
    - Se guardan su **nombre**, **especialidad** (como pesas, cardio, yoga, etc.) y su **nivel de experiencia**.
3. **Plan de Membresía**:
    - Cada plan tiene un **ID único**.
    - Los planes incluyen un **nombre del plan** (por ejemplo, "Mensual Básico", "Anual Premium"), una **duración en meses**, y el **costo**.
4. **Sesión**:
    - Cada sesión tiene un **ID único**, una **fecha y hora de inicio**, y una **duración en minutos**.
    - Un cliente puede reservar varias sesiones, pero cada sesión tiene solo un cliente y un entrenador asignado.
5. **Relaciones**:
    - Cada **Cliente** puede tener uno o más **Planes de Membresía**. Un cliente puede suscribirse a múltiples planes a lo largo del tiempo.
    - Cada **Cliente** puede reservar varias **Sesiones**.
    - Cada **Sesión** es atendida por un solo **Entrenador**.

Modela este sistema y asegúrate de reflejar correctamente las relaciones y atributos.

Solución

Modelo ER:

```
@startuml
entity "Cliente" {
    +ID_Cliente : int
    Nombre : string
    Correo : string
    Fecha_Registro : date
    Telefono : string
}

entity "Entrenador" {
    +ID_Entrenador : int
    Nombre : string
    Especialidad : string
    Nivel_Experiencia : string
}

entity "Plan_Membresia" {
    +ID_Plan : int
    Nombre_Plan : string
    Duracion_Meses : int
    Costo : float
}

entity "Sesion" {
    +ID_Sesion : int
    Fecha_Hora_Inicio : datetime
    Duracion_Minutos : int
}

entity "ClientePlan" {
    +ID_Cliente : int
    +ID_Plan : int
    Fecha_Inicio : date
}

' Relaciones entre entidades
Cliente ||--o{ Sesion : Reserva
Sesion }|--|| Entrenador : Atendida_por
Cliente ||--|{ ClientePlan : Suscripcion
ClientePlan }|--|| Plan_Membresia : Incluye
@enduml
```

Explicación

- La **entidad Cliente** contiene información sobre cada cliente registrado en el gimnasio.

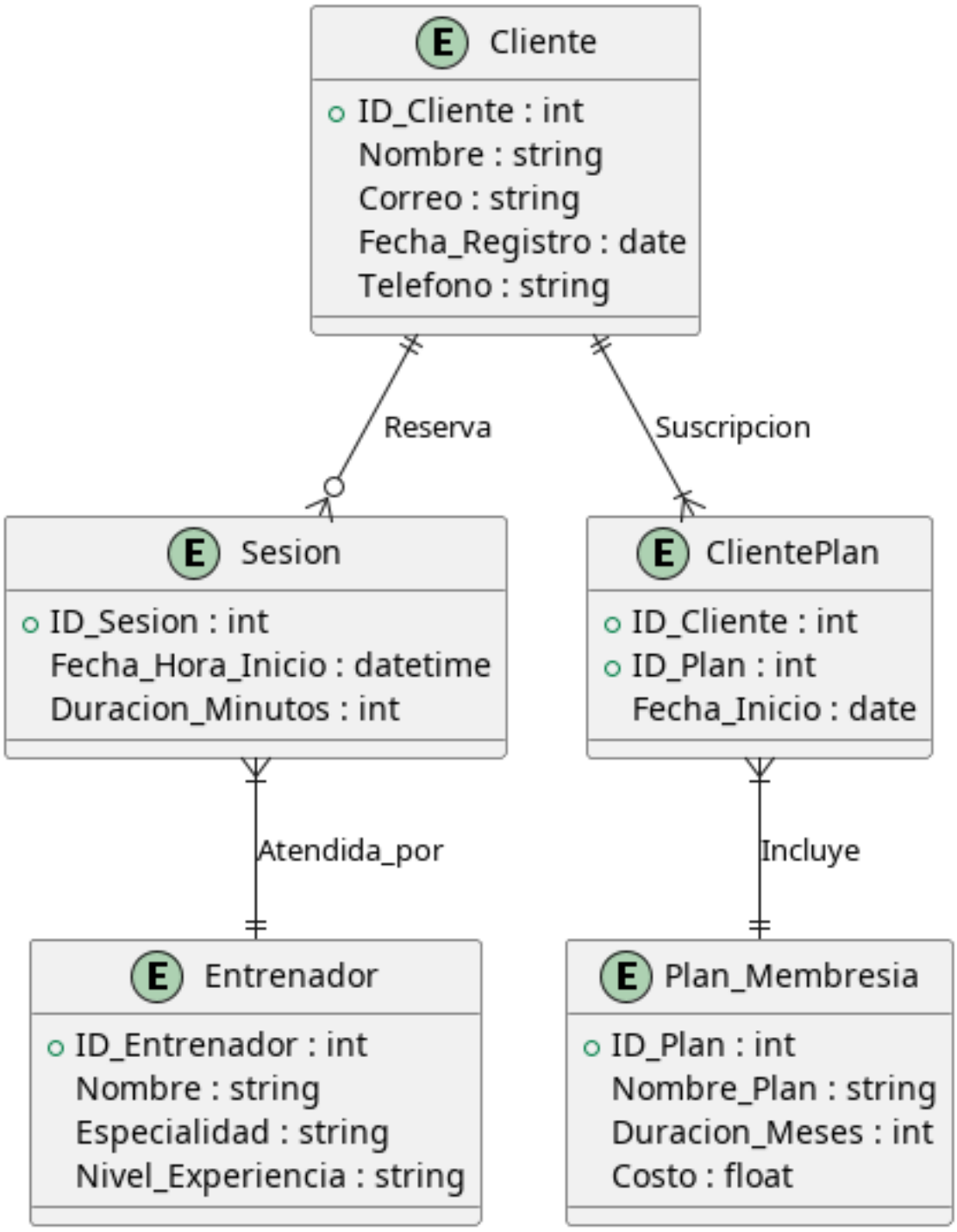

Figura 3: Diagrama Endidad Relación

- La **entidad Entrenador** almacena datos sobre los entrenadores, incluidos sus nombres, especialidades y experiencia.
- La **entidad Plan_Membresia** representa los diferentes planes de suscripción que un cliente puede elegir.
- La **entidad Sesion** representa cada sesión reservada en el gimnasio.
- La **entidad ClientePlan** es una entidad de unión para rastrear la suscripción de cada cliente a planes específicos, permitiendo múltiples registros a lo largo del tiempo.
-

Proyectos software

Una empresa de desarrollo de software necesita un sistema para organizar su información sobre **Proyectos**, **Empleados**, **Tareas** asignadas, **Equipos** de trabajo y **Tecnologías** utilizadas. Las especificaciones son las siguientes:

1. **Proyecto**:
    - Cada proyecto tiene un **ID único**.
    - Se guarda el **nombre del proyecto**, la **fecha de inicio**, la **fecha de finalización** y el **presupuesto**.
    - Cada proyecto puede requerir varias **Tecnologías** (como Python, JavaScript, SQL, etc.).
2. **Empleado**:
    - Cada empleado tiene un **ID único**.
    - Se guardan su **nombre**, **correo electrónico**, **puesto** (por ejemplo, desarrollador, gerente de proyecto, diseñador), y **fecha de contratación**.
3. **Equipo**:
    - Cada equipo tiene un **ID único**.
    - Cada equipo tiene un **nombre** y un **líder de equipo**.
    - Un equipo puede trabajar en múltiples proyectos a la vez, y cada proyecto puede tener varios equipos.
4. **Tarea**:
    - Cada tarea tiene un **ID único** y está vinculada a un proyecto.
    - Se registran el **nombre de la tarea**, la **descripción**, la **fecha límite** y el **estado** (pendiente, en progreso, completado).
    - Cada tarea debe asignarse a un **Empleado** específico.
5. **Tecnología**:

- Cada tecnología tiene un **ID único** y un **nombre** (como Java, Python, SQL).

6. **Relaciones**:
   - Cada **Proyecto** puede requerir varias **Tecnologías**, y cada **Tecnología** puede usarse en múltiples **Proyectos**.
   - Cada **Empleado** pertenece a un único **Equipo**, y cada **Equipo** puede incluir múltiples **Empleados**.
   - Cada **Tarea** pertenece a un **Proyecto** específico y está asignada a un único **Empleado**.
   - Cada **Equipo** puede trabajar en varios **Proyectos** a la vez, y cada **Proyecto** puede tener varios **Equipos** asignados.

Solución

Solución al problema:

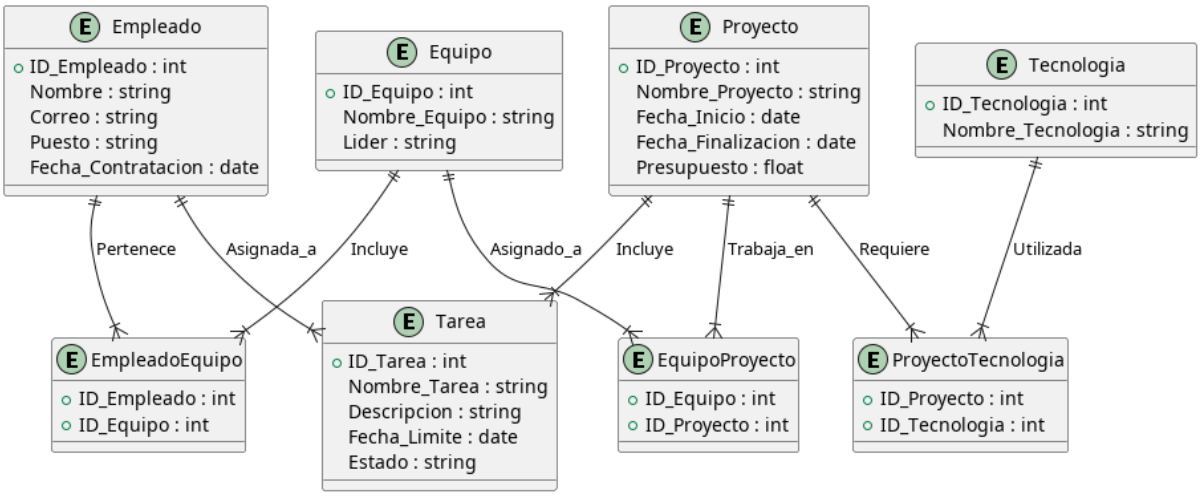

Figura 4: Diagrama Entidad Relación

```
@startuml
entity "Proyecto" {
    +ID_Proyecto : int
    Nombre_Proyecto : string
    Fecha_Inicio : date
    Fecha_Finalizacion : date
    Presupuesto : float
}

entity "Empleado" {
    +ID_Empleado : int
    Nombre : string
```

```
    Correo : string
    Puesto : string
    Fecha_Contratacion : date
}

entity "Equipo" {
    +ID_Equipo : int
    Nombre_Equipo : string
    Lider : string
}

entity "Tarea" {
    +ID_Tarea : int
    Nombre_Tarea : string
    Descripcion : string
    Fecha_Limite : date
    Estado : string
}

entity "Tecnologia" {
    +ID_Tecnologia : int
    Nombre_Tecnologia : string
}

entity "ProyectoTecnologia" {
    +ID_Proyecto : int
    +ID_Tecnologia : int
}

entity "EquipoProyecto" {
    +ID_Equipo : int
    +ID_Proyecto : int
}

entity "EmpleadoEquipo" {
    +ID_Empleado : int
    +ID_Equipo : int
}

' Relaciones entre entidades
Proyecto ||--|{ ProyectoTecnologia : Requiere
Tecnologia ||--|{ ProyectoTecnologia : Utilizada
Equipo ||--|{ EquipoProyecto : Asignado_a
Proyecto ||--|{ EquipoProyecto : Trabaja_en
Equipo ||--|{ EmpleadoEquipo : Incluye
Empleado ||--|{ EmpleadoEquipo : Pertenece
Proyecto ||--|{ Tarea : Incluye
Empleado ||--|{ Tarea : Asignada_a
@enduml
```

Explicación

- **Proyecto** contiene información sobre los proyectos de la empresa.
- **Empleado** almacena datos sobre los empleados, incluidos sus puestos y fechas de contratación.
- **Equipo** representa los equipos que trabajan en diferentes proyectos y puede tener un líder de equipo.
- **Tarea** representa cada tarea dentro de un proyecto y está asignada a un empleado.
- **Tecnologia** contiene las tecnologías que la empresa utiliza.
- **ProyectoTecnologia** es una tabla de unión que representa la relación de muchos-a-muchos entre **Proyecto** y **Tecnología**.
- **EquipoProyecto** representa la relación de muchos-a-muchos entre **Equipo** y **Proyecto**.
- **EmpleadoEquipo** representa la relación de muchos-a-muchos entre **Empleado** y **Equipo**.