TECNOLÓGICO
NACIONAL DE MÉXICO

Tecnológico Nacional de México, Campus Mexicali
Ingeniería en Sistemas Computacionales
Alumnos:
Emanuel Padilla Valencia
23490363
Clara Andrea Martinez Valdez
23490379
Luis Alonso Guevara Quiñonez
23490377

Materia: Fundamentos de Bases de Datos
Docente: José Ramón Bogarin Valenzuela
Tarea Unidad 2 Diagrama Modelo E-R y Diagrama de Venn
Mexicali Baja California, al 24 de febrero del 2024.

1 Plataforma de Comercio Electrónico 

Identificación de entidades

Las principales entidades de la plataforma son:

- **Usuario**: Representa a los clientes que realizan compras.
- **Producto**: Representa los artículos disponibles en la tienda.
- **Pedido**: Registra las órdenes de compra de los usuarios.
- **DetallePedido**: Relaciona los productos con los pedidos e indica la cantidad comprada.

Definición de atributos clave

- **Usuario**: ID_Usuario (PK), Nombre, Correo, Dirección.
- **Producto**: ID_Producto (PK), Nombre, Precio, Stock.
- **Pedido**: ID_Pedido (PK), ID_Usuario (FK), Fecha, Total.
- **DetallePedido**: ID_Detalle (PK), ID_Pedido (FK), ID_Producto (FK), Cantidad, Subtotal.

Relaciones entre entidades

- Un **Usuario** puede realizar múltiples **Pedidos**.
- Un **Pedido** puede incluir múltiples **Productos** a través de **DetallePedido**.
- Un **Producto** puede aparecer en varios **Pedidos**, pero con diferentes cantidades.

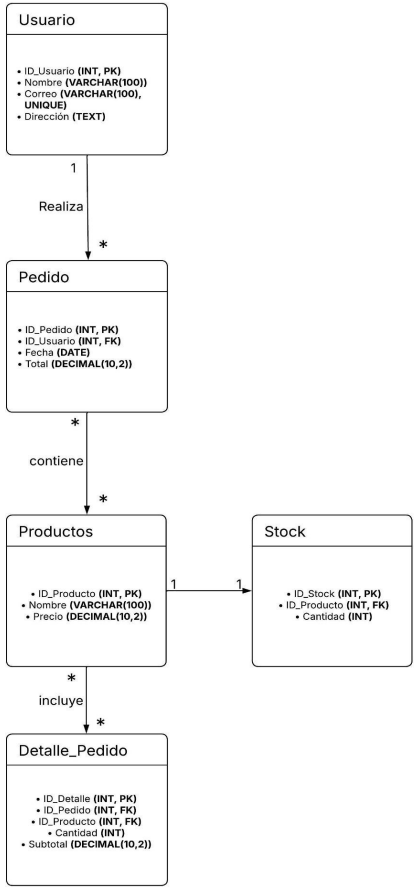
Claves primarias y foráneas

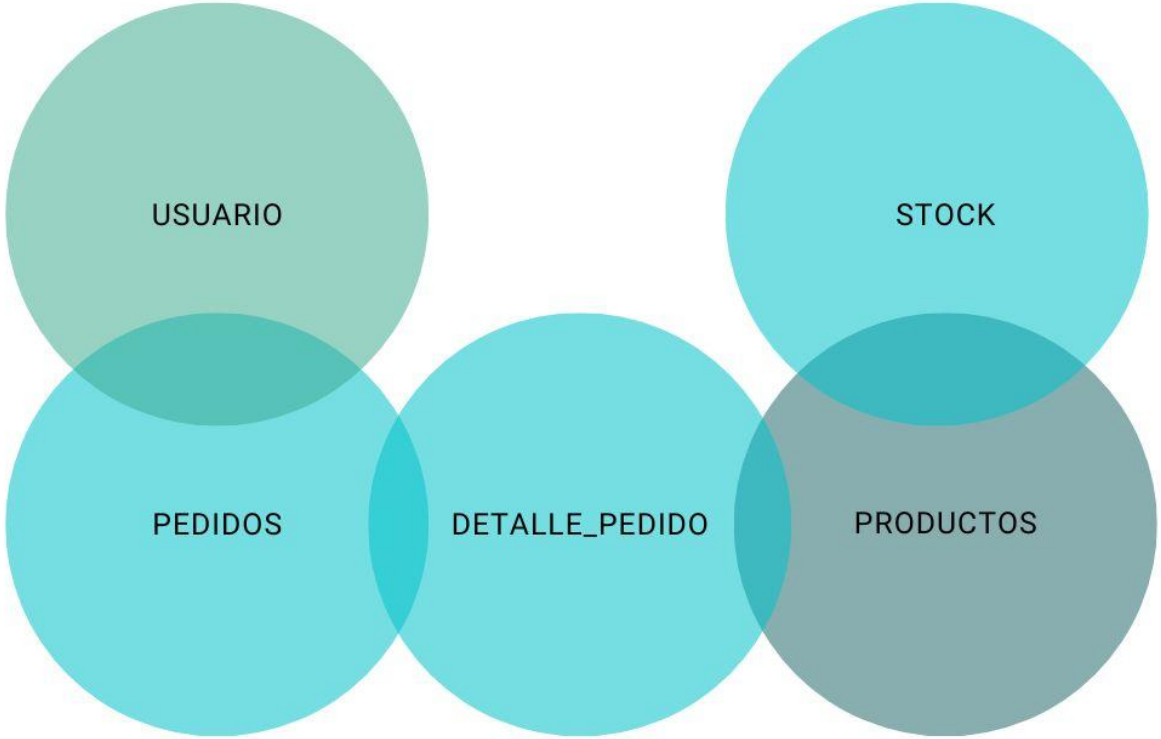
- **ID_Usuario** es clave primaria en Usuario.
- **ID_Producto** es clave primaria en Producto.
- **ID_Pedido** es clave primaria en Pedido.
- **ID_Detalle** es clave primaria en DetallePedido.
- **ID_Usuario** en Pedido es clave foránea referenciando a Usuario.
- **ID_Pedido** e **ID_Producto** en DetallePedido son claves foráneas referenciando a Pedido y Producto.

Refinamiento del diseño

- **Normalización**: Evita la redundancia de información al separar productos y pedidos en entidades independientes.
- **Reglas de negocio**: Un producto no puede ser comprado si su stock es insuficiente.
- **Índices**: Se recomienda indexar **Correo** en Usuario y **Nombre** en Producto para búsquedas rápidas.

DIAGRAMA MODELO E-R
Y DIAGRAMA DE VENN





Creación de la Base de Datos en PostgreSQL

```sql
CREATE TABLE Usuario (
    ID_Usuario SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Correo VARCHAR(100) UNIQUE NOT NULL,
    Dirección TEXT
);

CREATE TABLE Producto (
    ID_Producto SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Precio DECIMAL(10,2) NOT NULL
);

CREATE TABLE Stock (
    ID_Stock SERIAL PRIMARY KEY,
    ID_Producto INT UNIQUE NOT NULL,
    Cantidad INT NOT NULL CHECK (Cantidad >= 0),
    FOREIGN KEY (ID_Producto) REFERENCES Producto(ID_Producto) ON DELETE CASCADE
);

CREATE TABLE Pedido (
    ID_Pedido SERIAL PRIMARY KEY,
    ID_Usuario INT NOT NULL,
    Fecha DATE NOT NULL DEFAULT CURRENT_DATE,
    Total DECIMAL(10,2) NOT NULL CHECK (Total >= 0),
    FOREIGN KEY (ID_Usuario) REFERENCES Usuario(ID_Usuario) ON DELETE CASCADE
);

CREATE TABLE DetallePedido (
    ID_Detalle SERIAL PRIMARY KEY,
    ID_Pedido INT NOT NULL,
    ID_Producto INT NOT NULL,
    Cantidad INT NOT NULL CHECK (Cantidad > 0),
    Subtotal DECIMAL(10,2) NOT NULL CHECK (Subtotal >= 0),
    FOREIGN KEY (ID_Pedido) REFERENCES Pedido(ID_Pedido) ON DELETE CASCADE,
    FOREIGN KEY (ID_Producto) REFERENCES Producto(ID_Producto) ON DELETE CASCADE
);
```

Conclusión

Este modelo E-R garantiza un diseño eficiente y estructurado de la plataforma de comercio electrónico, asegurando la correcta gestión de pedidos, productos y usuarios. La implementación del sistema optimiza el almacenamiento de datos, mantiene la integridad de la información y mejora el control del stock y la facturación.

2 Sistema de Gestión Escolar 

Identificación de entidades

Las principales entidades del sistema son:

- **Alumno:** Representa a los estudiantes que se inscriben en los cursos.
- **Curso**: Representa los cursos disponibles en el sistema.
- **Profesor**: Representa a los profesores que imparten las materias.
- **Inscripción**: Registra la inscripción de un alumno en un curso.
- **Calificación**: Almacena las calificaciones de los alumnos en los cursos.

Definición de atributos clave

- **Alumno:**ID_Alumno (PK), Nombre, Correo, Teléfono
- **Curso:** ID_Curso(PK), Nombre_Curso, Descripción, ID_Profesor(FK)
- **Profesor:** ID_Profesor(PK), Nombre, Correo, Teléfono.
- **Inscripción:** ID_Inscripción (PK), ID_Alumno (FK), ID_Curso (FK), Fecha_Inscripcion.
- **Calificación:** ID_Calificación (PK), ID_Inscripcion (FK), Nota; Fecha_Calificación.

Relaciones entre entidades

- Un **Alumno** puede inscribirse en múltiples **Cursos.**
- Un **Curso** puede ser impartido por un único **Profesor**.
- Un **Profesor** puede impartir múltiples **Cursos.**
- Una **Inscripción** está asociada a un único **Alumno** y a un único Curso.
- Una **Calificación** está asociada a una única **Inscripción.**

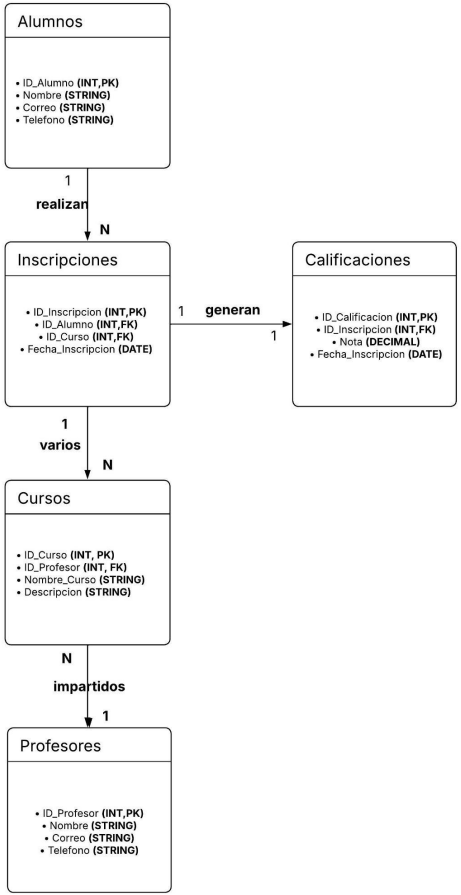
Claves primarias para identificación única

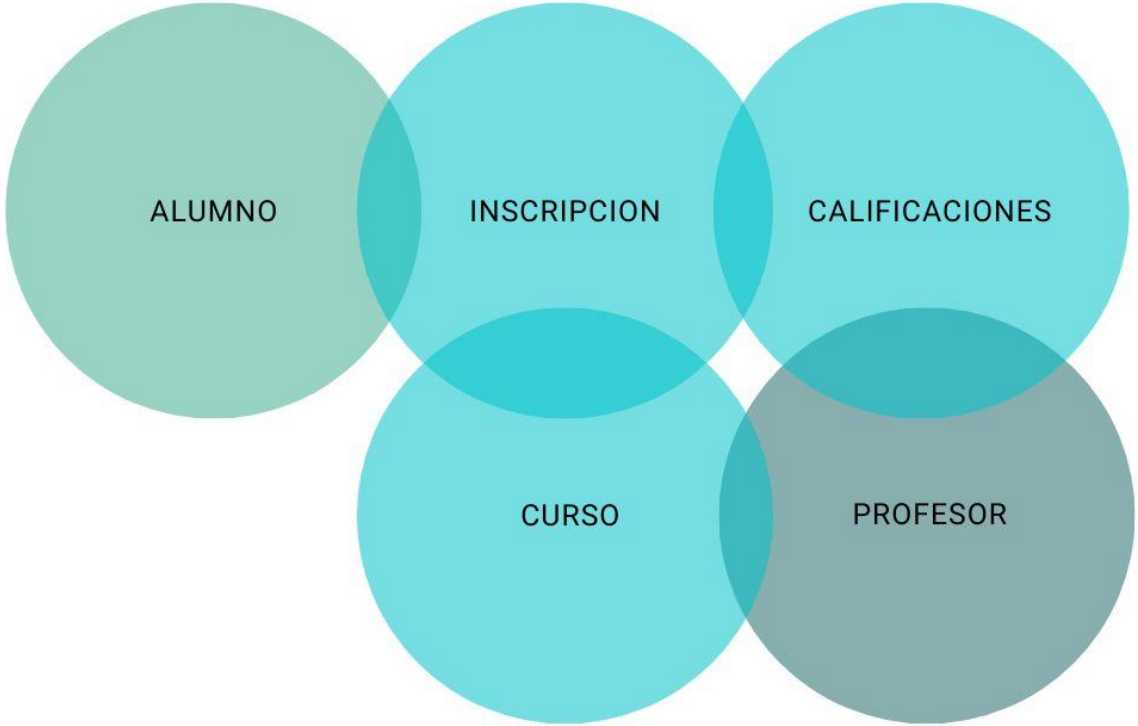
- **ID_Alumno** como clave primaria de Alumno.
- **ID_Curso** como clave primaria de Curso.
- **ID_Profesor** como clave primaria de Profesor.
- **ID_Inscripcion** como clave primaria de Inscripción.
- **ID_Calificación** como clave primaria de Calificación.

Refinamiento del diseño

- **Normalización:** Se separan los datos en entidades bien definidas para evitar redundancia. Por ejemplo, los datos del alumno no se duplican en cada inscripción.
- **Reglas de negocio:** Se establece una restricción para evitar que en un alumno se inscriba dos veces en el mismo curso.

DIAGRAMA MODELO E-R
Y DIAGRAMA DE VENN





Creación de la Base de Datos en PostgreSQL

```sql
CREATE TABLE Alumno (
    ID_Alumno SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Correo VARCHAR(100) UNIQUE NOT NULL,
    Teléfono VARCHAR(15)
);

CREATE TABLE Profesor (
    ID_Profesor SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Correo VARCHAR(100) UNIQUE NOT NULL,
    Teléfono VARCHAR(15)
);

CREATE TABLE Curso (
    ID_Curso SERIAL PRIMARY KEY,
    Nombre_Curso VARCHAR(100) NOT NULL,
    Descripción TEXT,
    ID_Profesor INT REFERENCES Profesor(ID_Profesor) ON DELETE CASCADE
);

CREATE TABLE Inscripción (
    ID_Inscripción SERIAL PRIMARY KEY,
    ID_Alumno INT REFERENCES Alumno(ID_Alumno) ON DELETE CASCADE,
    ID_Curso INT REFERENCES Curso(ID_Curso) ON DELETE CASCADE,
    Fecha_Inscripción DATE NOT NULL,
    CONSTRAINT inscripción_unica UNIQUE (ID_Alumno, ID_Curso)
);

CREATE TABLE Calificación (
    ID_Calificación SERIAL PRIMARY KEY,
    ID_Inscripción INT REFERENCES Inscripción(ID_Inscripción) ON DELETE CASCADE,
    Nota DECIMAL(5,2) NOT NULL,
    Fecha_Calificación DATE NOT NULL
);
```

Conclusión

Este modelo E-R garantiza un diseño eficiente y estructurado del sistema de gestión escolar, asegurando la correcta administración de inscripciones, cursos y calificaciones. La organización de las entidades y sus relaciones optimiza el almacenamiento de datos y mejora la gestión académica, permitiendo un seguimiento preciso del desempeño de los alumnos y la asignación de materias a los profesores.

3 Aplicación de Mensajería 

Identificación de entidades

Las principales entidades del sistema de mensajería son:

- **Usuario**: Representa a cada persona que puede enviar y recibir mensajes.
- **Mensaje**: Contiene el contenido del mensaje y la información de envío.
- **Adjunto**: Representa los archivos enviados junto con un mensaje.

Definición de atributos clave

- **Usuario**: ID_Usuario (PK), Nombre, Correo, Contraseña
- **Mensaje**: ID_Mensaje (PK), ID_Emisor (FK), ID_Receptor (FK), Contenido, Fecha_Hora
- **Adjunto**: ID_Adjunto (PK), ID_Mensaje (FK), Nombre_Archivo, Tipo_Archivo, Tamaño

Establecer relaciones

- Un **usuario** puede enviar y recibir múltiples **mensajes**.
- Un **mensaje** pertenece a un emisor y un receptor.
- Un **mensaje** puede tener múltiples **adjuntos**, pero cada adjunto pertenece a un solo mensaje.

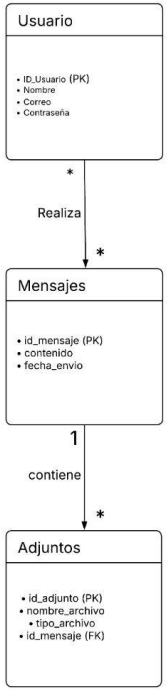
Elección de claves primarias y foráneas

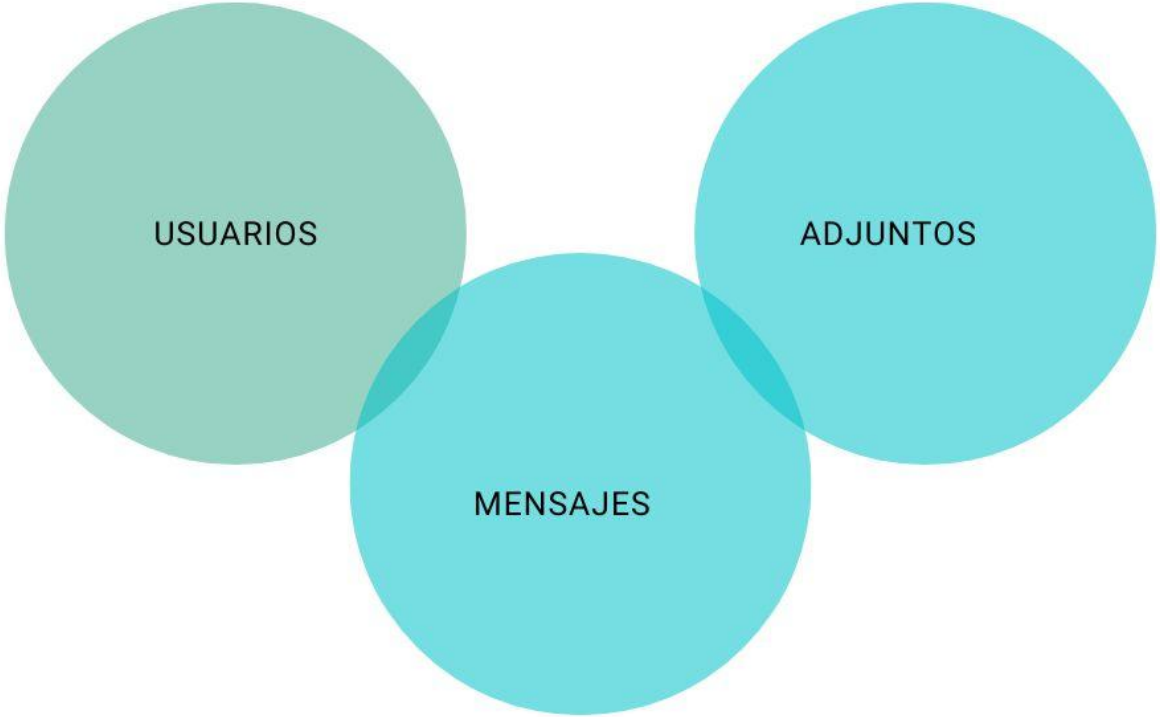
- `ID_Usuario` como clave primaria en Usuario.
- `ID_Mensaje` como clave primaria en Mensaje.
- `ID_Adjunto` como clave primaria en Adjunto.
- `ID_Emisor` y `ID_Receptor` en Mensaje referencian `ID_Usuario`.
- `ID_Mensaje` en Adjunto referencia `ID_Mensaje` en Mensaje.

Refinamiento del diseño

- Normalización: Se evitan redundancias al separar **mensajes** y **adjuntos** en tablas distintas.
- Índices: Se puede indexar `ID_Usuario` y `Fecha_Hora` en **Mensaje** para búsquedas eficientes.

DIAGRAMA MODELO E-R
Y DIAGRAMA DE VENN





Creación de la Base de Datos en PostgreSQL

```sql
CREATE TABLE Usuario (
    id_usuario SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL
);

CREATE TABLE Mensaje (
    id_mensaje SERIAL PRIMARY KEY,
    contenido TEXT NOT NULL,
    fecha_envio TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Adjunto (
    id_adjunto SERIAL PRIMARY KEY,
    nombre_archivo VARCHAR(255) NOT NULL,
    tipo_archivo VARCHAR(50) NOT NULL,
    id_mensaje INT NOT NULL,
    CONSTRAINT fk_mensaje FOREIGN KEY (id_mensaje) REFERENCES Mensaje(id_mensaje) ON DELETE CASCADE
);

CREATE TABLE Mensaje_Usuario (
    id_usuario_emisor INT NOT NULL,
    id_usuario_receptor INT NOT NULL,
    id_mensaje INT NOT NULL,
    PRIMARY KEY (id_usuario_emisor, id_usuario_receptor, id_mensaje),
    CONSTRAINT fk_usuario_emisor FOREIGN KEY (id_usuario_emisor) REFERENCES Usuario(id_usuario) ON DELETE CASCADE,
    CONSTRAINT fk_usuario_receptor FOREIGN KEY (id_usuario_receptor) REFERENCES Usuario(id_usuario) ON DELETE CASCADE,
    CONSTRAINT fk_mensaje_usuario FOREIGN KEY (id_mensaje) REFERENCES Mensaje(id_mensaje) ON DELETE CASCADE
);
```

Conclusión

Este modelo E-R garantiza un diseño eficiente y estructurado de la aplicación de mensajería, asegurando la correcta gestión de usuarios, mensajes y archivos adjuntos. La organización del historial de conversaciones y las relaciones entre entidades optimizan el almacenamiento de datos y mejoran la experiencia del usuario en la plataforma.

4 Plataforma de Streaming de música 

Identificar las entidades del sistema

Las principales entidades del sistema son:

- **Usuario:** Representa a los usuarios que escuchan música y crean playlists.
- **Canción:** Representa las canciones subidas por los artistas.
- **Playlist:** Contiene canciones añadidas por los usuarios.
- **HistorialReproducción:** Almacena las reproducciones de canciones por cada usuario.

Definición de atributos clave

- **Usuario:** ID_Usuario, Nombre, Correo, Suscripción.
- **Canción:** ID_Cancion, Titulo, ID_Artista, Duración, Género.
- **Playlist:** ID_Playlist, ID_Usuario, Nombre, Fecha_Creación.
- **HistorialReproduccion:** ID_Historial, ID_Usuario, ID_Canción, Fecha_Reproduccion.

Relaciones entre entidades

- ★ Un **Usuario** puede tener varias **Playlists**.
- ★ Una **Playlist** puede contener varias **Canciones** (relación muchos a muchos).
- ★ Un **Usuario** puede escuchar varias **Canciones**, y estas se almacenan en el **Historial de Reproducción**.
- ★ Un **Artista** puede subir varias **Canciones**.

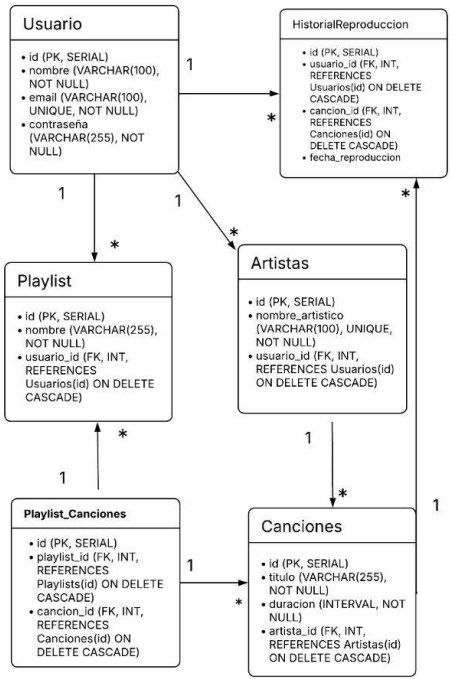
Claves primarias y foráneas

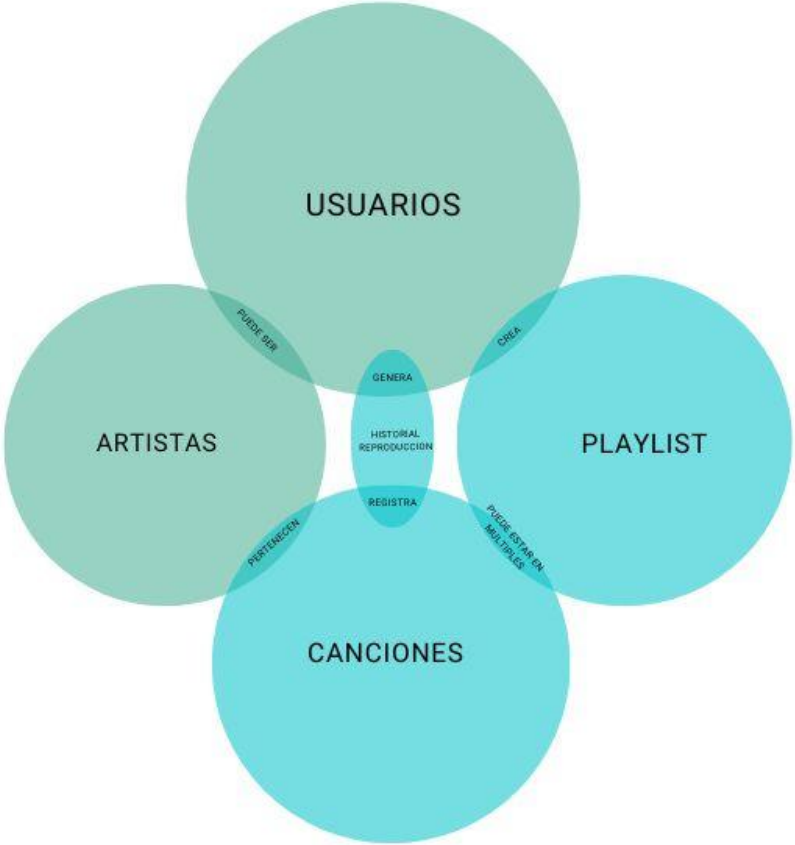
- ID_Usuario (INT, PK)
- ID_Canción (INT, PK)
- ID_Artista (INT, FK) → Usuario(ID_Usuario)`
- ID_Playlist (INT, PK)
- ID_Usuario (INT, FK) → Usuario(ID_Usuario)`
- ID_Historial (INT, PK)
- ID_Usuario (INT, FK) → Usuario(ID_Usuario)`
- ID_Canción (INT, FK) → Canción(ID_Canción)`

Refinamiento del diseño

Normalización: Separar datos en entidades distintas para evitar redundancia.
Reglas de negocio: Un usuario solo puede agregar una canción una vez a una playlist.

DIAGRAMA MODELO E-R Y DIAGRAMA DE VENN





Creación de la Base de Datos en PostgreSQL

```sql
CREATE TABLE Usuarios (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    contraseña VARCHAR(255) NOT NULL
);

CREATE TABLE Artistas (
    id SERIAL PRIMARY KEY,
    nombre_artistico VARCHAR(100) UNIQUE NOT NULL,
    usuario_id INT REFERENCES Usuarios(id) ON DELETE CASCADE
);

CREATE TABLE Canciones (
    id SERIAL PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    duracion INTERVAL NOT NULL,
    artista_id INT REFERENCES Artistas(id) ON DELETE CASCADE
);

CREATE TABLE Playlists (
    id SERIAL PRIMARY KEY,
    nombre VARCHAR(255) NOT NULL,
    usuario_id INT REFERENCES Usuarios(id) ON DELETE CASCADE
);

CREATE TABLE Playlist_Canciones (
    id SERIAL PRIMARY KEY,
    playlist_id INT REFERENCES Playlists(id) ON DELETE CASCADE,
    cancion_id INT REFERENCES Canciones(id) ON DELETE CASCADE,
    UNIQUE (playlist_id, cancion_id)
);

CREATE TABLE HistorialReproduccion (
    id SERIAL PRIMARY KEY,
    usuario_id INT REFERENCES Usuarios(id) ON DELETE CASCADE,
    cancion_id INT REFERENCES Canciones(id) ON DELETE CASCADE,
    fecha_reproduccion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Conclusión

Este modelo E-R garantiza un diseño eficiente y estructurado de la plataforma de streaming de música, asegurando la correcta gestión de usuarios, playlists y canciones. La organización del historial de reproducción y la relación entre entidades optimizan el almacenamiento de datos y mejoran la experiencia del usuario en el sistema.

Sistema de Gestión de Citas Médicas 🏥

1. Identificar las entidades del sistema

Las entidades principales son:

- **Pacientes**: Personas que solicitan citas médicas.
- **Médicos**: Profesionales que atienden las citas.
- **Citas**: Registros de las citas agendadas.
- **Especialidades**: Áreas médicas en las que se especializan los médicos.

2. Definir atributos clave para cada entidad

Pacientes

- **ID_Paciente** (PK)
- Nombre, Apellido, Fecha_Nacimiento, Género, Teléfono, Correo, Dirección

Médicos

- **ID_Medico** (PK)
- Nombre, Apellido, Especialidad (FK), Teléfono, Correo, Horario_Disponible

Citas

- **ID_Cita** (PK)
- Fecha, Hora, Estado (pendiente, atendida, cancelada), **ID_Paciente** (FK), **ID_Medico** (FK)

Especialidades

- **ID_Especialidad** (PK)
- Nombre_Especialidad

Establecer relaciones entre entidades

- **Pacientes** y **Citas**: Un paciente puede tener múltiples citas, pero cada cita pertenece a un solo paciente (relación 1:N).
- **Médicos** y **Citas**: Un médico puede tener múltiples citas, pero cada cita es atendida por un solo médico (relación 1:N).
- **Médicos** y **Especialidades**: Un médico puede tener una sola especialidad, pero una especialidad puede ser asignada a múltiples médicos (relación 1:N).

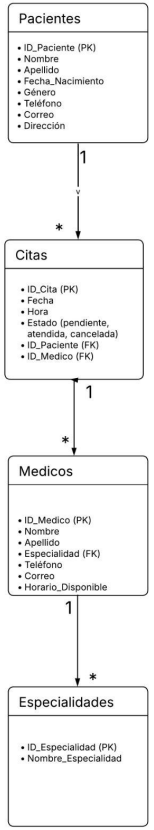
Elegir claves primarias para identificación única

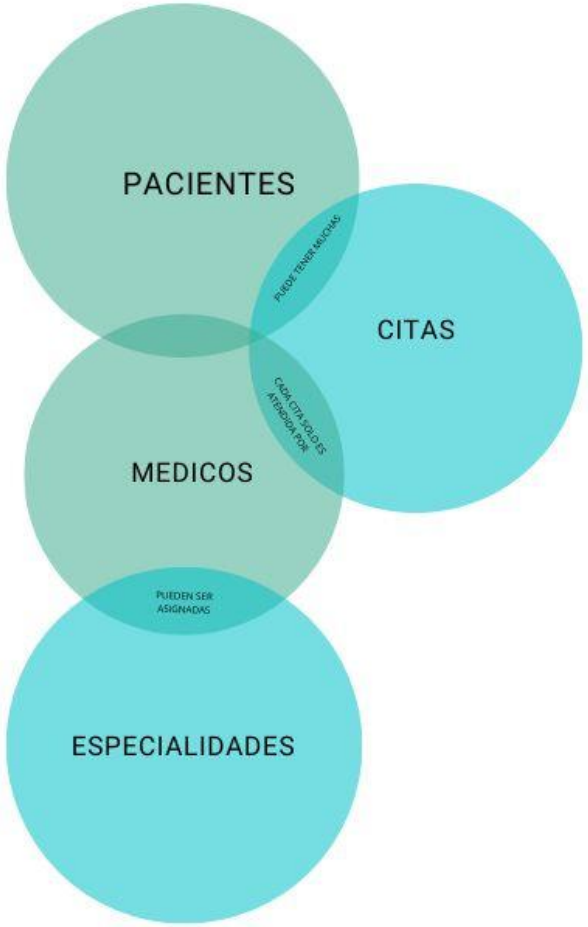
- **Pacientes**: `ID_Paciente`
- **Médicos**: `ID_Medico`
- **Citas**: `ID_Cita`
- **Especialidades**: `ID_Especialidad`

Refinar el diseño para optimizar la estructura

- Agregar restricciones para evitar citas duplicadas en la misma fecha y hora para un médico.
- Validar que el estado de la cita solo pueda ser "pendiente", "atendida" o "cancelada".
- Asegurar que los horarios de los médicos no se solapen.
- Normalizar la base de datos para evitar redundancias.

DIAGRAMA MODELO E-R
Y DIAGRAMA DE VENN





Creación de la Base de Datos en PostgreSQL

```sql
CREATE TABLE Especialidades (
    ID_Especialidad SERIAL PRIMARY KEY,
    Nombre_Especialidad VARCHAR(100) NOT NULL
);

CREATE TABLE Medicos (
    ID_Medico SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Apellido VARCHAR(100) NOT NULL,
    Especialidad INT REFERENCES Especialidades(ID_Especialidad) ON DELETE CASCADE,
    Telefono VARCHAR(15),
    Correo VARCHAR(100),
    Horario_Disponible TEXT
);

CREATE TABLE Pacientes (
    ID_Paciente SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Apellido VARCHAR(100) NOT NULL,
    Fecha_Nacimiento DATE,
    Genero CHAR(1),
    Telefono VARCHAR(15),
    Correo VARCHAR(100),
    Direccion TEXT
);

CREATE TABLE Citas (
    ID_Cita SERIAL PRIMARY KEY,
    Fecha DATE NOT NULL,
    Hora TIME NOT NULL,
    Estado VARCHAR(20) CHECK (Estado IN ('pendiente', 'atendida', 'cancelada')),
    ID_Paciente INT REFERENCES Pacientes(ID_Paciente) ON DELETE CASCADE,
    ID_Medico INT REFERENCES Medicos(ID_Medico) ON DELETE CASCADE
);

CREATE TABLE Horarios_Medicos (
    ID_Horario SERIAL PRIMARY KEY,
    ID_Medico INT REFERENCES Medicos(ID_Medico) ON DELETE CASCADE,
    Hora_Inicio TIME NOT NULL,
    Hora_Fin TIME NOT NULL
);
```

Conclusión

Este modelo E-R garantiza un diseño eficiente y estructurado del sistema de gestión de citas médicas, asegurando la correcta administración de pacientes, médicos y horarios. La organización de las citas con sus respectivos estados optimiza el almacenamiento de datos y mejora la gestión del servicio, permitiendo un seguimiento preciso de la disponibilidad y atención médica.

6 Biblioteca Digital 

Identificar las entidades del sistema

Las entidades principales son:

- **Usuarios**: Personas que pueden tomar libros en préstamo.
- **Libros**: Los libros disponibles en la biblioteca.
- **Préstamos**: Registros de los préstamos de libros.

Definir atributos clave para cada entidad

Usuarios

- ID_Usuario (PK)
- Nombre
- Apellido
- Correo
- Tipo_Usuario (normal, administrador)

Libros

- ID_Libro (PK)
- Título
- Autor
- ISBN
- Disponible (booleano: true o false)

Préstamos

- ID_Prestamo (PK)
- Fecha_Prestamo
- Fecha_Devolucion_Esperada
- Fecha_Devolucion_Real
- ID_Usuario (FK)
- ID_Libro (FK)

Elegir claves primarias para identificación única

- **Usuarios**: `ID_Usuario`
- **Libros**: `ID_Libro`
- **Préstamos**: `ID_Prestamo`

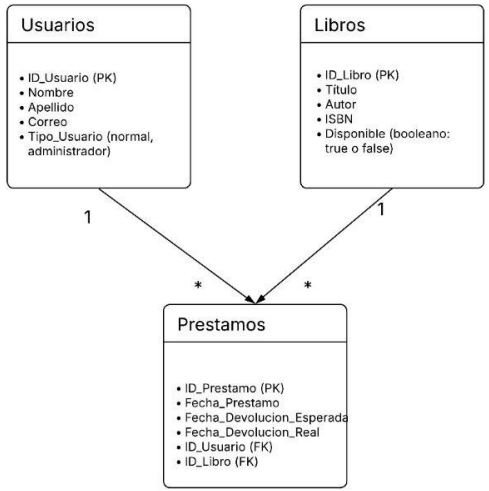
Establecer relaciones entre entidades

- **Usuarios** y **Préstamos**: Un usuario puede tener muchos préstamos, pero cada préstamo pertenece a un solo usuario (relación 1:N).
- **Libros** y **Préstamos**: Un libro puede estar en muchos préstamos, pero cada préstamo corresponde a un solo libro (relación 1:N).

Refinar el diseño para optimizar la estructura

- Asegurar que solo los usuarios administradores puedan gestionar los libros (esto se manejará en la lógica de la aplicación).
- Validar que un libro no pueda prestarse si no está disponible.
- Agregar restricciones para evitar préstamos duplicados para el mismo libro y usuario.

DIAGRAMA MODELO E-R Y DIAGRAMA DE VENN



Creación de la Base de Datos en PostgreSQL

```sql
CREATE TABLE Usuarios (
    ID_Usuario SERIAL PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Apellido VARCHAR(100) NOT NULL,
    Correo VARCHAR(100) UNIQUE NOT NULL,
    Tipo_Usuario VARCHAR(20) CHECK (Tipo_Usuario IN ('normal', 'administrador'))
);

CREATE TABLE Libros (
    ID_Libro SERIAL PRIMARY KEY,
    Titulo VARCHAR(200) NOT NULL,
    Autor VARCHAR(100) NOT NULL,
    ISBN VARCHAR(20) UNIQUE NOT NULL,
    Disponible BOOLEAN DEFAULT TRUE
);

CREATE TABLE Prestamos (
    ID_Prestamo SERIAL PRIMARY KEY,
    Fecha_Prestamo DATE NOT NULL,
    Fecha_Devolucion_Esperada DATE NOT NULL,
    Fecha_Devolucion_Real DATE,
    ID_Usuario INT REFERENCES Usuarios(ID_Usuario) ON DELETE CASCADE,
    ID_Libro INT REFERENCES Libros(ID_Libro) ON DELETE CASCADE
);
```

Conclusiones

Esta E-R garantiza una estructura muy simple y detallada de información otorgandonos un sistema optimo de gestión en una biblioteca digital.

7 Sistema de Gestión de Proyectos 📊

Identificación de entidades

Las principales entidades del sistema de gestión de proyectos son:

- **Empresas:** Representan a las organizaciones que registran proyectos en el sistema.
- **Proyectos:** Representan los trabajos que las empresas gestionan.
- **Tareas:** Representan las actividades que vienen dentro de un proyecto.
- **Usuarios:** Representan a las personas que interactúan con tareas que son implementadas en proyectos.

Definición de atributos clave para cada entidad

- **Empresas**: id_empresa (PK), nombre, dirección, teléfono
- **Proyectos**: id_proyecto (PK), nombre, descripción, fecha_inicio, fecha_fin, id_empresa (FK),id_Usuario(FK).
- **Tareas**: id_tarea (PK), nombre, descripción, estado, id_proyecto (FK),id_usuario(FK).
- **Usuarios**: id_usuario (PK), nombre, email, puesto

Relaciones entre entidades

- Un **empresa** puede registrar múltiples **proyectos.** (uno a muchos)
- Un **usuario** puede ser responsable de varias **tareas** (uno a muchos)
- Un **proyecto** tiene varias **tareas**. (uno a muchos)

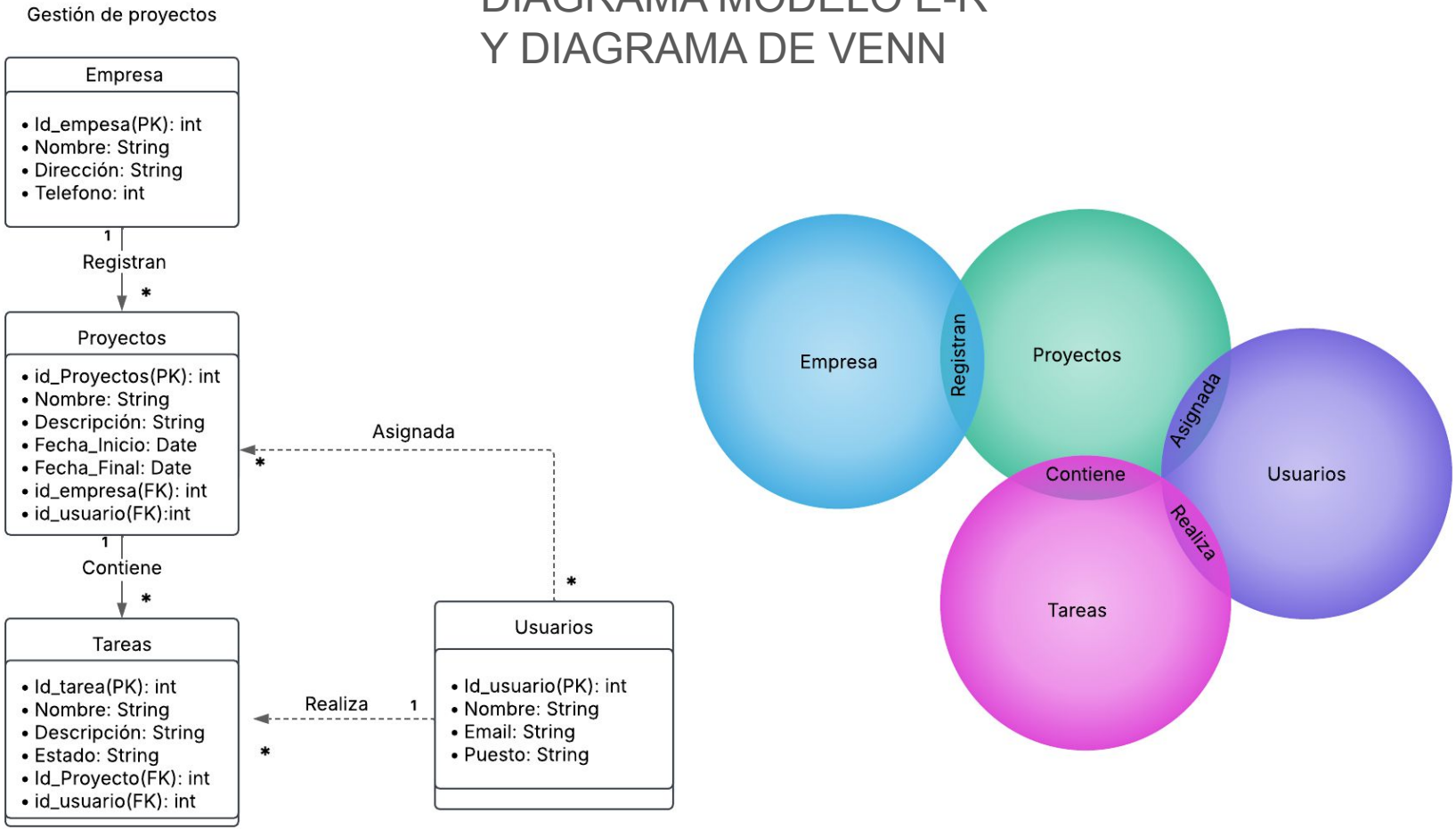
Claves primarias y foráneas

- **id_empresa(PK)** como clave primaria en Empresa.
- **id_proyecto(PK)** como clave primaria en Proyecto.
- **id_tarea(PK)** como clave primaria en Tarea.
- **id_usuario(PK)** como clave primaria en Usuario.
- **id_empresa(FK)** en Proyecto referencia a Empresa.
- **id_proyecto(FK)** en Tarea referencia a Proyecto.
- **id_usuario(FK)** En Proyecto y Tarea referencia a Usuario.

Refinamiento del diseño

- **Normalización:** Se eliminan grupos repetitivos y se aseguran valores atómicos.
- **Reglas del negocio:** Un usuario puede solo trabajar en proyectos de su empresa.

DIAGRAMA MODELO E-R
Y DIAGRAMA DE VENN



Query | Query History

Creación de la Base de Datos en PostgreSQL

```sql
CREATE TABLE Empresa (
    id_empresa SERIAL PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL,
    direccion TEXT NOT NULL,
    telefono INT NOT NULL
);

CREATE TABLE Usuarios (
    id_usuario SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    puesto VARCHAR(30) NOT NULL
);

CREATE TABLE Proyectos (
    id_proyecto SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    fecha_inicio DATE NOT NULL,
    fecha_final DATE NOT NULL,
    id_empresa INT NOT NULL,
    id_usuario INT NOT NULL,
    FOREIGN KEY (id_empresa) REFERENCES Empresa(id_empresa) ON DELETE CASCADE,
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario) ON DELETE SET NULL
);

CREATE TABLE Tareas (
    id_tarea SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    estado VARCHAR(30) NOT NULL CHECK (estado IN ('Pendiente', 'En Progreso', 'Completada')),
    id_proyecto INT NOT NULL,
    id_usuario INT NOT NULL,
    FOREIGN KEY (id_proyecto) REFERENCES Proyectos(id_proyecto) ON DELETE CASCADE,
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario) ON DELETE SET NULL
);
```

Conclusión

Esta E-R nos asegura un sistema de gestión de proyectos óptimo, donde la integridad de los datos y optimización se aplican, permitiendonos una buena gestión, otorgandonos así la satistacción tanto de la empresa como de los empleados.

8 Red Social 📱

Identificación de entidades

Las principales entidades del sistema de red social son:

- **Usuarios:** Representan a las personas que comentaran, reaccionaran y publicaran.
- **Publicaciones:** Representan lo que pueden subir los usuarios.
- **Comentarios:** Representan los pensamientos de los usuarios con las publicaciones.
- **Reacciones:** Representan las interacciones de los usuarios con las publicaciones.

Definición de atributos clave para cada entidad

- **Usuarios:** id_usuario(PK), nombre, email, fecha_registro.
- **Publicaciones:** id_publicacion(PK), id_usuario(FK), contenido, fecha_publicación.
- **Comentarios:** id_comentario(PK), id_usuario(FK), id_publicacion(FK), contenido, fecha_comentario
- **Reacciones:** id_reaccion(PK), id_usuario(FK), id_publicacion(FK), id_comentario(FK), tipo_reaccion, fecha_reaccion.
- **Seguidores(Tabla intermedia):** id_seguidor(FK usuario), id_seguido(FK usuario), fecha_seguimiento.

Relaciones entre entidades

- Un **usuario** puede hacer muchas **publicaciones.** (uno a muchos)
- Un **usuario** puede hacer muchos comentarios en **publicaciones**. (uno a muchos)
- Un **usuario** puede reaccionar a muchas **publicaciones** y **comentarios**. (uno a muchos)

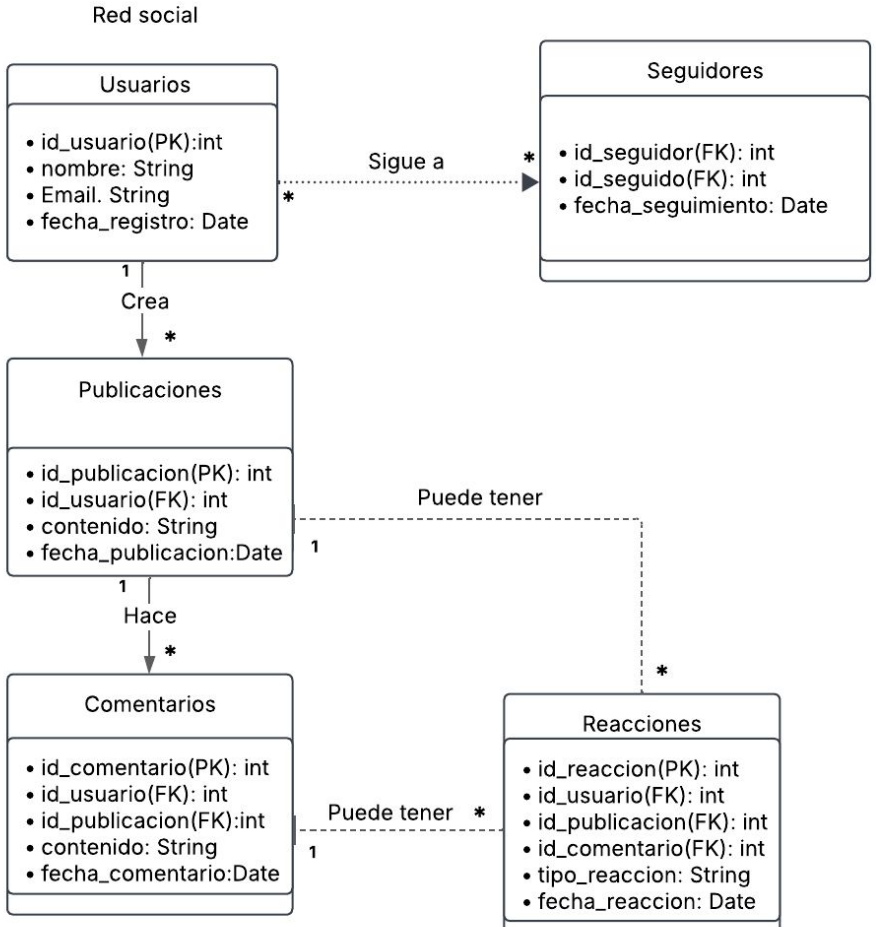
Claves primarias y foráneas

- **id_usuario(PK)** como clave primaria en Usuario.
- **id_publicacion(PK)** como clave primaria en Publicación:
- **id_comentario(PK)** como clave primaria en Comentario.
- **id_reaccion(PK)** como clave primaria en reacción.
- **id_usuario(FK)** en Publicacion, Comentario y Reacción referencia a Usuario.
- **id_publicacion(FK)** en Comentario y Reacción referencia a Publicación.
- **id_comentario(FK)** en Reacción referencia a Comentario.
- **id_seguidor(FK)** e **id_seguido(FK)** en Seguidores referencia a Usuario.

Refinamiento del diseño

- **Normalización:** Todos los atributos dependen completamente de la clave primaria (En Reacciones, tipo_reaccion depende solo de id_reaccion).
- **Reglas del negocio:** No permitir que un usuario se siga a si mismo.

DIAGRAMA MODELO E-R Y DIAGRAMA DE VENN



Creación de la Base de Datos en PostgreSQL

Query | Query History

```sql
CREATE TABLE Usuarios (
    id_usuario SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    fecha_registro DATE NOT NULL DEFAULT CURRENT_DATE
);

CREATE TABLE Seguidores (
    id_seguidor INT NOT NULL,
    id_seguido INT NOT NULL,
    fecha_seguimiento DATE NOT NULL DEFAULT CURRENT_DATE,
    PRIMARY KEY (id_seguidor, id_seguido),
    FOREIGN KEY (id_seguidor) REFERENCES Usuarios(id_usuario) ON DELETE CASCADE,
    FOREIGN KEY (id_seguido) REFERENCES Usuarios(id_usuario) ON DELETE CASCADE
);

CREATE TABLE Publicaciones (
    id_publicacion SERIAL PRIMARY KEY,
    id_usuario INT NOT NULL,
    contenido TEXT NOT NULL,
    fecha_publicacion DATE NOT NULL DEFAULT CURRENT_DATE,
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario) ON DELETE CASCADE
);

CREATE TABLE Comentarios (
    id_comentario SERIAL PRIMARY KEY,
    id_usuario INT NOT NULL,
    id_publicacion INT NOT NULL,
    contenido TEXT NOT NULL,
    fecha_comentario DATE NOT NULL DEFAULT CURRENT_DATE,
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario) ON DELETE CASCADE,
    FOREIGN KEY (id_publicacion) REFERENCES Publicaciones(id_publicacion) ON DELETE CASCADE
);

CREATE TABLE Reacciones (
    id_reaccion SERIAL PRIMARY KEY,
    id_usuario INT NOT NULL,
    id_publicacion INT NULL,
    id_comentario INT NULL,
    tipo_reaccion VARCHAR(50) NOT NULL CHECK (tipo_reaccion IN ('Like', 'Amor', 'Jaja', 'Wow', 'Triste', 'Enojado')),
    fecha_reaccion DATE NOT NULL DEFAULT CURRENT_DATE,
    FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario) ON DELETE CASCADE,
    FOREIGN KEY (id_publicacion) REFERENCES Publicaciones(id_publicacion) ON DELETE CASCADE,
    FOREIGN KEY (id_comentario) REFERENCES Comentarios(id_comentario) ON DELETE CASCADE
);
```

Conclusión

Esta E-R fue diseñada para que nos garantice una interacción más adecuada a los usuarios y en el entorno, permitiendo que puedan tener seguidores, puedan seguir a mas gente y reaccionando a comentarios así también como publicaciones.

9 Sistema de facturación 💸

Identificación de entidades

Las principales entidades del sistema de facturación son:

- **Clientes:** Representan a las personas que compraran los productos y recibiran las facturas.
- **Factura:** Represetan lo que se genera al comprar un producto.
- **DetalleFactura:** Representan los datos que contiene la factura de la compra.
- **Productos:** Represetan lo que van a comprar los usuarios.

Definición de atributos clave para cada entidad

- **Clientes**: id_cliente,(PK), nombre, email, telefono, dirección
- **Facturas**: id_factura(PK), id_cliente(FK), fecha_creacion, total.
- **DetalleFactura**: id_detalle(PK), id_factura(FK), id_producto(FK), cantidad, precio, descuento
- **Producto**: id_producto(PK), nombre, precio, impuesto

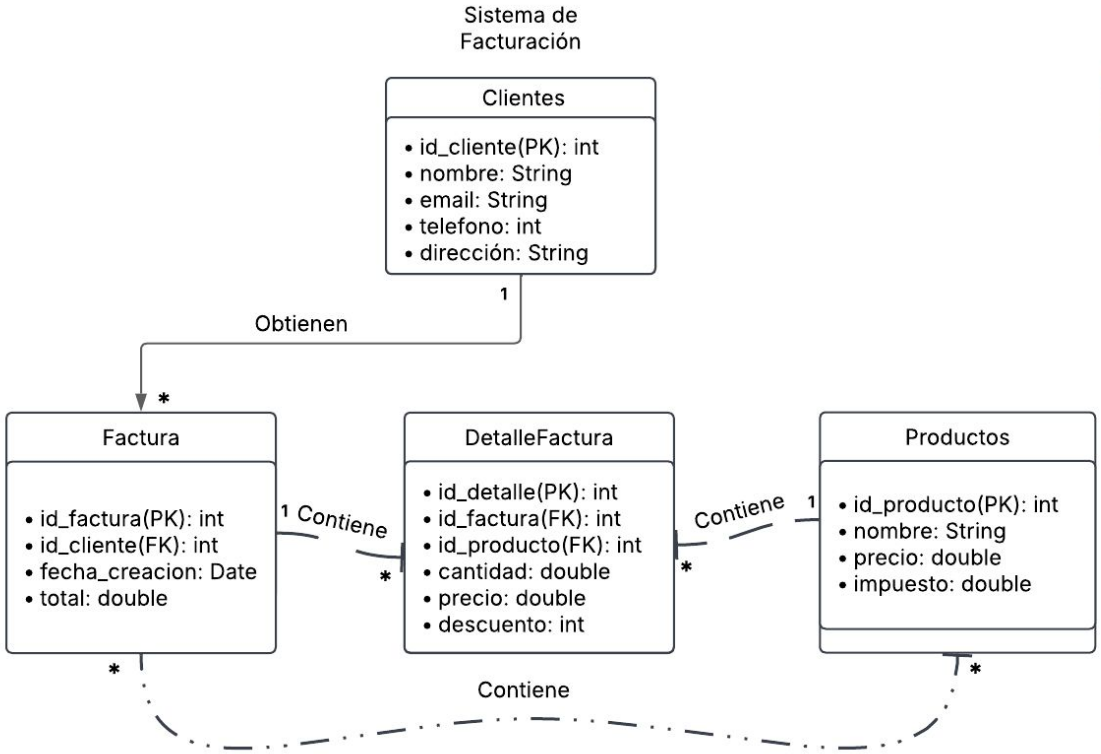
Relaciones entre entidades

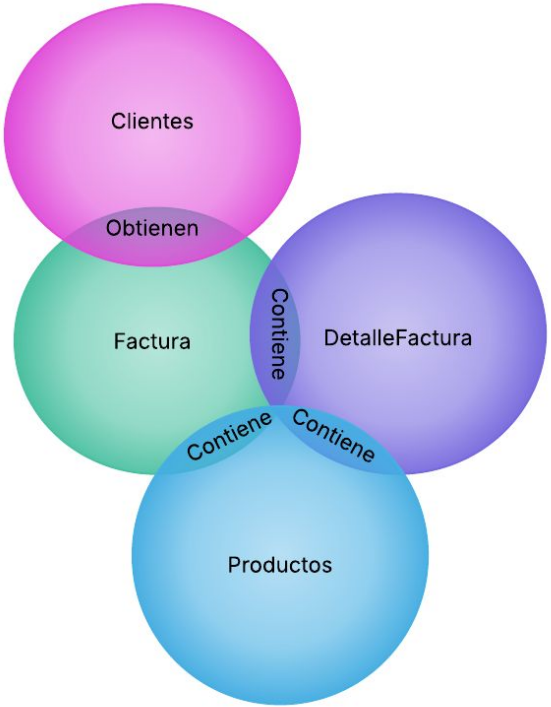
- Un **Cliente** puede tener muchas **Facturas.** (uno a muchos)
- Una **Factura** puede tener muchos **Detalles** de **Factura**. (uno a muchos)
- Un **producto** puede aparecer en multiples **DetallesFactura**. (uno a muchos)

Claves primarias y foráneas

- **id_cliente** como clave primaria en **Cliente.**
- **id_factura** como clave primaria en **Factura.**
- **id_detalle** como clave primaria en **DetalleFactura.**
- **id_producto** como clave primaria en **Producto.**
- **id_cliente** en **Factura** referencia a **Cliente.**
- **id_factura** en **DetalleFactura** referencia a **Factura.**
- **id_producto** en **DetalleFactura** referencia a **Producto.**

DIAGRAMA MODELO E-R Y DIAGRAMA DE VENN





Refinamiento del diseño

- **Normalización:** Se evita redundancia almacenando los productos en tablas separadas
- **Reglas del negocio:** No permitir facturas sin detalles de compra.

Query | Query History

```sql
CREATE TABLE Clientes (
    id_cliente SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    telefono BIGINT NOT NULL,
    direccion TEXT NOT NULL
);

CREATE TABLE Factura (
    id_factura SERIAL PRIMARY KEY,
    id_cliente INT NOT NULL,
    fecha_creacion DATE NOT NULL DEFAULT CURRENT_DATE,
    total DOUBLE PRECISION NOT NULL DEFAULT 0.0,
    FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente) ON DELETE CASCADE
);

CREATE TABLE Productos (
    id_producto SERIAL PRIMARY KEY,
    nombre VARCHAR(70) NOT NULL,
    precio DOUBLE PRECISION NOT NULL CHECK (precio >= 0),
    impuesto DOUBLE PRECISION NOT NULL CHECK (impuesto >= 0)
);

CREATE TABLE DetalleFactura (
    id_detalle SERIAL PRIMARY KEY,
    id_factura INT NOT NULL,
    id_producto INT NOT NULL,
    cantidad DOUBLE PRECISION NOT NULL CHECK (cantidad > 0),
    precio DOUBLE PRECISION NOT NULL CHECK (precio >= 0),
    descuento INT NOT NULL CHECK (descuento >= 0),
    FOREIGN KEY (id_factura) REFERENCES Factura(id_factura) ON DELETE CASCADE,
    FOREIGN KEY (id_producto) REFERENCES Productos(id_producto) ON DELETE CASCADE
```

Creación de la Base de Datos en PostgreSQL

Conclusión

Este modelo E-R garantiza un ingreso de información óptimo donde cada tabla esta ordenada y conectada de manera eficiente permitiendo así una mejor estructura en un sistema de facturación.