BASE DE DATOS PARA UN SISTEMA DE MENSAJERIA.

Contexto del proyecto:

En este proyecto pongo a prueba mis conocimientos sobre lo que son las bases de datos en lenguaje SQL y PL/SQL para una nueva aplicación de mensajería.

Se necesita una base de datos relacional que soporte todas las funcionalidades clave de la aplicación, incluyendo:

- Gestión de usuarios.
- Gestión de dispositivos.
- Gestión de mensajes.
- Gestión de conversaciones.
- Gestión de contactos.
- Y más.

A continuación, se detallan todo lo implementado para este proyecto, desde los requerimientos funcionales, las reglas del negocio, búsqueda de entidades con sus respectivos atributos, relaciones entre entidades, mapeo, modelo entidad relación ER, código en SQL, implementación de PL/SQL para el sistema.

Reglas del Negocio:

1. Unicidad:
    a. El correo electrónico deben ser únicos.
2. Seguridad:
    a. Las contraseñas deben almacenarse de forma segura.
    b. Los tokens de acceso deben ser únicos y expirar después de un cierto periodo de tiempo.
3. Consistencia de Datos:
    a. Un usuario solo puede participar una vez en cada conversación.
    b. Los mensajes deben pertenecer a una única conversación y ser enviados por un único usuario.
4. Reglas de Bloqueo:
    a. Un usuario no puede enviar mensajes a otro usuario que lo haya bloqueado.
    b. Los bloqueos deben ser reversibles.

Requerimientos funcionales:

1. Gestión de Usuarios:
    a. Registro de usuarios con detalles personales (nombre de usuario, correo electrónico, contraseña, apellidos, número de teléfono).
    b. Posibilidad s activar o desactivar usuarios, así como bloqueo de usuarios.
2. Gestión de Dispositivos:

    a. Cada usuario puede registrar múltiples dispositivos con un ID de dispositivo único y tipo.
    b. Almacenamiento de un token único por dispositivo.
3. Acceso a Usuarios:
    a. Generación de tokens de acceso cuando un usuario inicia sesión desde un dispositivo.
    b. Almacenamiento del historial de accesos, incluyendo el ID del dispositivo y la fecha de creación.
4. Gestión de Contactos:
    a. Los usuarios pueden gestionar una lista de contactos con información básica.
5. Conversación y participantes:
    a. Los usuarios pueden iniciar y participar en conversaciones.
    b. Las conversaciones deben tener un título, un creador y un canal asociado.
    c. Registro de la fecha de creación y última actualización de cada conversación.
    d. Almacenamiento de los participantes por cada conversación, indicando el tipo de participante.
6. Mensajes:
    a. Envío de mensajes dentro de las conversaciones.
    b. Almacenamiento del texto de mensajes, tipo de mansaje (texto, imagen, video) URLs de archivos adjuntos.
    c. Registro de la fecha de creación y eliminación de mensajes.
7. Reportes:
    a. Los usuarios deben poder reportar a otros usuarios o mensajes.
    b. Almacenamiento de información del reporte, incluyendo notas y fecha de creación.
8. Gestión de Eliminaciones:
    a. Almacenamiento de registros de conversaciones y mensajes eliminados, con la fecha de eliminación y el usuario que realizo la acción.
9. Lista de bloqueo:
    a. Los usuarios deben poder bloquear a otros usuarios.
    b. Almacenamiento de información sobre quien bloquea a quien y cuando se realizó la acción.

Requerimientos Técnicos:

1. Utilizar PostgreSQL como gestor de base de datos.
2. Implementación de procedimientos almacenados para las operaciones CRUD en cada una de las tablas.
3. Garantizar la integridad y referencia y la normalización de los datos.

ENTIDADES Y RELACIONES.

- Usuarios:
  - Contiene la información de los usuarios.
- Dispositivos:
  - Relacionados con los usuarios, contiene la información de los dispositivos.
- Conversaciones:
  - Contiene la información sobre las conversaciones.
- Participantes:
  - Relaciona usuarios con conversaciones.
- Mensajes:
  - Contiene la información de los mensajes enviados en las conversaciones.
- Contactos:
  - Contiene información de los contactos de los usuarios.
- Reportes:
  - Contiene información de los reportes hechos por los usuarios.
- Acceso:
  - Gestiona los tokens de acceso de los usuarios.
- Bloqueo de usuarios:
  - Contiene información sobre usuarios bloqueados.
- Contacto del usuario:
  - Relaciona usuarios con sus contactos.
- Conversaciones eliminadas:
  - Contiene información sobre las conversaciones eliminadas.
- Mensajes eliminados:
  - Contiene información sobre los mensajes eliminados.

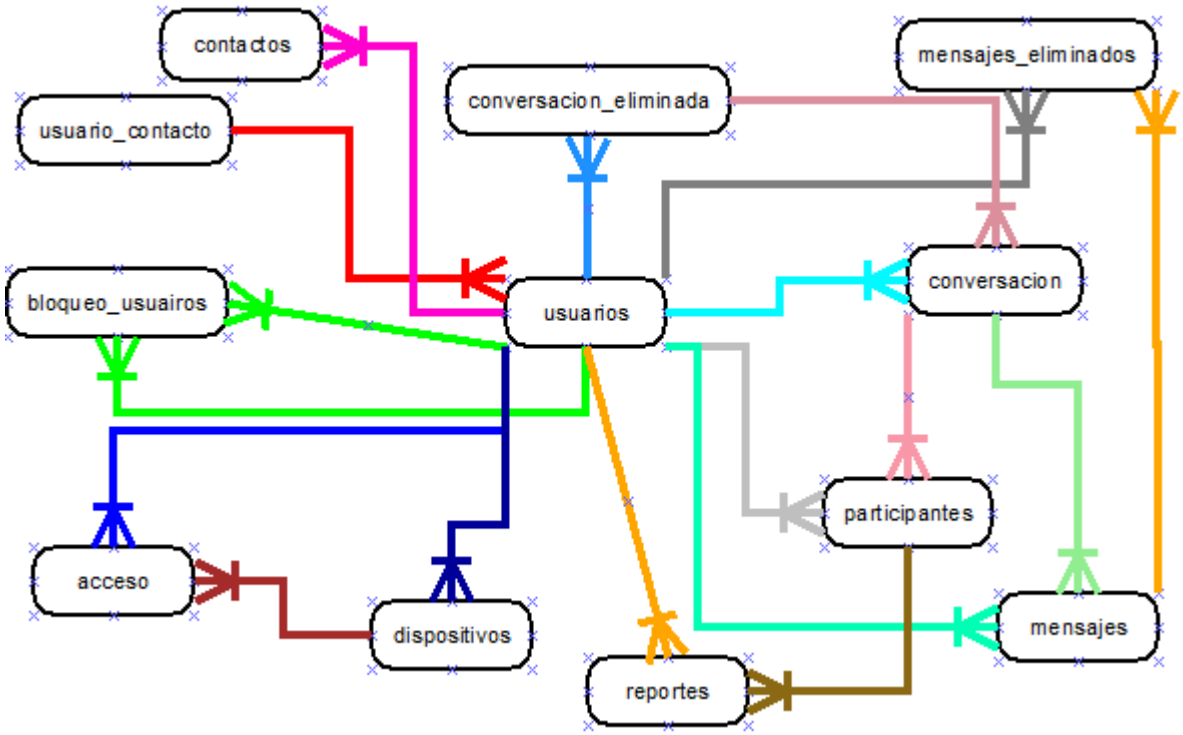
RELACIONES ENTRE ENTIDADES.

- usuarios ↔ dispositivos:
  - Un usuario puede tener múltiples dispositivos.
- usuarios ↔ accesos:
  - Un usuario puede tener múltiples accesos.
- usuarios ↔ lista_bloqueados:
  - Un usuario puede bloquear a múltiples usuarios.
- usuarios ↔ contactos:
  - Un usuario puede tener múltiples contactos.
- usuarios ↔ contacto_usuario:
  - Un usuario puede tener múltiples contactos específicos.
- usuarios ↔ conversaciones:
  - Un usuario puede crear múltiples conversaciones.
- conversaciones ↔ participantes:
  - Una conversación puede tener múltiples participantes.
- usuarios ↔ participantes:
  - Un usuario puede participar en múltiples conversaciones.

- conversaciones ↔ mensajes:
    - Una conversación puede tener múltiples mensajes.
- usuarios ↔ mensajes:
    - Un usuario puede enviar múltiples mensajes.
- mensajes ↔ mensajes_eliminados:
    - Un mensaje puede ser eliminado.
- conversaciones ↔ conversaciones_eliminadas:
    - Una conversación puede ser eliminada.
- usuarios ↔ reportes:
    - Un usuario puede realizar múltiples reportes.
- participantes ↔ reportes:
    - Un participante puede ser reportado múltiples veces.

PRIMER MODELO ER

Este es un primer modelo presentado solo haciendo las relaciones de las entidades que se obtuvieron, mostramos las relaciones en colores para poder identificar mejor como es que se relacionan las entidades.



MAPEO

ENTIDADES CON ATRIBUTOS Y REFERENCIAS.

Tabla Usuarios.

idUsuario	username	email	password	nombre	apellido
serial	varchar(255)	varchar(255)	varchar(255)	varchar(255)	varchar(255)
NN	NN	NN	NN	NN	NN

teléfono	Estado actividad	Estado bloqueo	Fecha creación	Fecha actualización
varchar(30)	bool	bool	time	time
NN	NN	NN	NN	NN

Tabla Dispositivos.

idDispositivo	idUsuario	tipoDispositivo	claveDispositivo
serial	int	varchar(255)	varchar(255)
NN	references usuarios(idUsuario)	NN	NN

Tabla Acceso

idAcceso	IdUsuario	idDispositivo	token	fecRegistro	fecActualizacion
serial	int	Int	varchar(100)	varchar(100)	varchar(100)
NN	references usuarios (idUsuario)	References dispositivos (idDispositivo)	NN	NN	NN

Tabla Usuarios_Bloqueados

idUsBlqueo	idUsuario	fechaBloqueo	telefono	idParticipante
serial	Int	Date	varchar()	int
NN	references usuarios (idUsuario)	NN	NN	references participantes (idParticipante)

Tabla Contactos

idContacto	nombre	teléfono	email	apellido	idUsuario
serial	varchar(255)	varchar(255)	varchar(255)	varchar(255)	int
NN	NN	NN	NN	NN	references usuarios (idUsuario)

Tabla Contacto_Usuairo

idUsContacto	idUsuario	nombre	fecCreacion	apellido
serial	int	varchar(255)	Timestamp	varchar(255)
NN	references usuarios (idUsuario)	NN	NN	NN

Tabla Conversación.

idConversacion	titulo	idUsuario	idCanal	fechCreacion
serial	varchar(255)	Int	varchar(255)	timestamp
NN	NN	references usuarios (idUsuario)	NN	NN

fechEliminacion	
timestamp	
NN	

Tabla Participantes.

idParticipante	idConversacion	idUsuario	tipoUsuario
serial	int	int	enum (‘user’, ‘admin’, ‘moderador’)
NN	references conversacion (idConversacion)	references usuarios (idUsuario)	NN

Tabla Mensajes.

idMensaje	idConversacion	idUsuario	tipoMensaje	mensaje
serial	int	int	Enum ('TEXT', 'IMAGE', 'VIDEO', 'FILE')	varchar(255)
NN	references conversacion (idConversacion)	references usuarios (idUsuario)	NN	NN

fechEnvio	fechEliminacion
timestamp	timestamp
NN	NN

Tabla Mensajes_Eliminados.

idMsj	idMensaje	idUsuario	fechEliminado
serial	int	int	timastamp
NN	references mensajes (idMensaje)	references usuarios (idUsuario)	NN

Tabla Conversacion_Eliminada.

idConv	idConversacion	idUsuario	fechElimimacion
serial	int	int	timestamp
NN	references conversacion (idConversacion)	references usuarios (idUsuario)	NN

Tabla Reportes.

idReporte	idUsuairo	idParticipante	tipoReporte	nota	fechReporte
serial	int	int	varchar(255)	varchar(255)	Timestamp
NN	references usuarios (idUsuario)	references participantes (idParticipante)	NN	NN	NN

NORMALIZACION DE LA BASE DE DATOS

Vamos a demostrar que la base de datos esta en las primeras formas normales (1FN, 2FN, 3FN), vamos a verificar y mostrar ejemplos que cumplan con los requisitos de cada forma.

Primera forma normal (1FN)

Ejemplo de la tabla usuarios en 1FN:

- Cada columna contiene valores atómicos.
- Todas las columnas tiene valores del mismo tipo.
- Cada columna tiene un nombre único.
- El orden de almacenamiento de los datos no importa.

Segunda forma normal (2FN)

Ejemplo de la tabla mensajes en 2FN:

- La tabla esta en 1FN
- Todos los atributos no primarios depende completamente de la clave primaria idMensaje.

Tercera forma normal (3FN)

Ejemplo de la tabla reportes en 3FN:

- La tabla esta en 2FN

- No hay dependencias transitivas. Todos los atributos no primarios dependen directamente de la clave primaria idReporte.

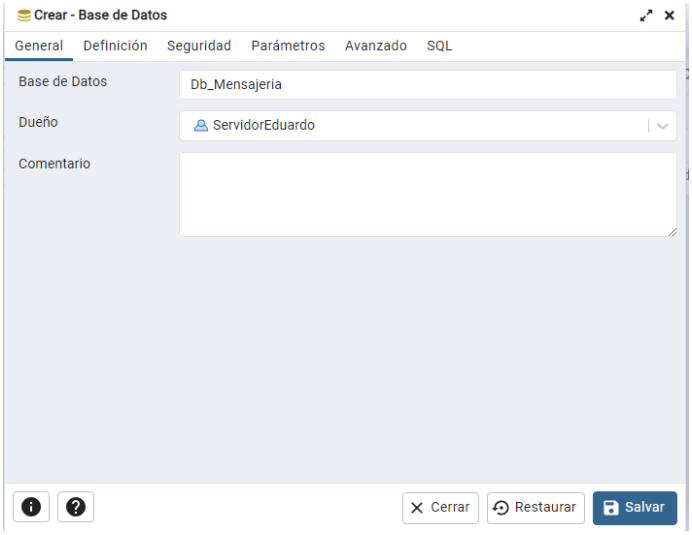
NOTA: En estos ejemplos demostramos como las tablas en la base de datos cumplen con las primeras formas normales. Cada tabla está estructurada de manera que evita redundancias y asegura la integridad de los datos.

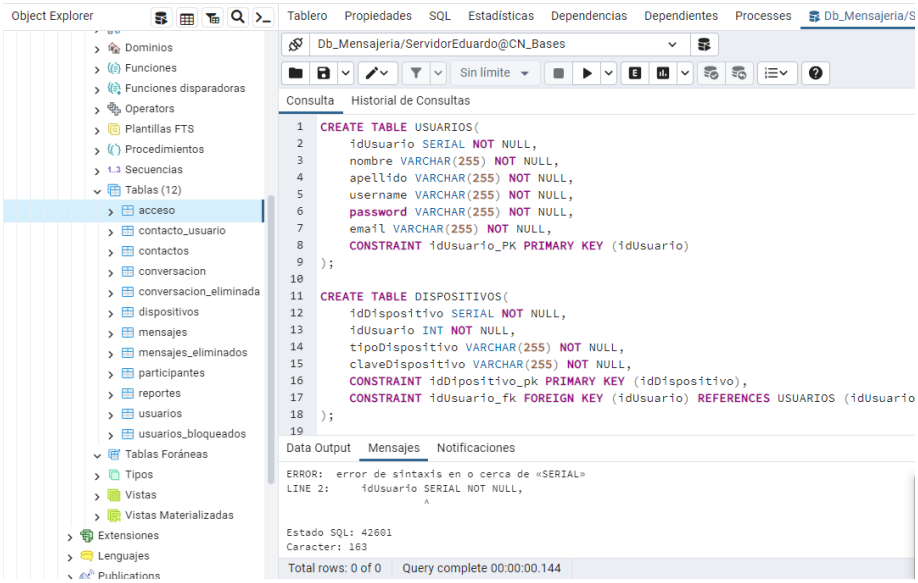
CODIGO SQL

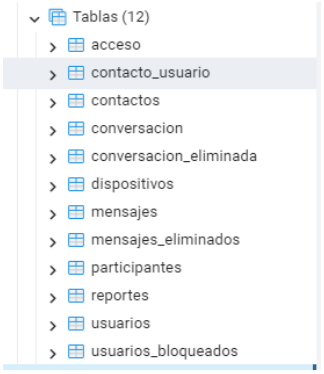
CREATE TABLE USUARIOS(idUsuario SERIAL NOT NULL, nombre VARCHAR(255) NOT NULL, apellido VARCHAR(255) NOT NULL, username VARCHAR(255) NOT NULL, password VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL, CONSTRAINT idUsuario_PK PRIMARY KEY (idUsuario));	CREATE TABLE DISPOSITIVOS(idDispositivo SERIAL NOT NULL, idUsuario INT NOT NULL, tipoDispositivo VARCHAR(255) NOT NULL, claveDispositivo VARCHAR(255) NOT NULL, CONSTRAINT idDipositivo_pk PRIMARY KEY (idDispositivo), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));
CREATE TABLE CONTACTOS(idContacto SERIAL NOT NULL, idUsuario INT NOT NULL, nombre VARCHAR(255) NOT NULL, apellido VARCHAR(255) NOT NULL, teléfono VARCHAR(255) NOT NULL, email VARCHAR(255) NOT NULL, CONSTRAINT idContacto_pk PRIMARY KEY (idContacto), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));	CREATE TABLE CONTACTO_USUARIO(idUsContacto SERIAL NOT NULL, idUsuario INT NOT NUULL, nombre VARCHAR(255) NOT NULL, apellido VARCHAR(255) NOT NULL, fecCreacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP, CONSTRAINT idUsContacto_pk PRIMARY KEY (idUsContacto), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));
CREATE TABLE CONVERSACION(idConversacion SERIAL NOT NULL, titulo VARCHAR(255) NOT NULL, idUsuario INT NOT NULL, idCanalVARCHAR(255) NOT NULL, fechCreacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP, fechEliminacion TIMESTAMP, CONSTRAINT idConversacion_pk PRIMARY KEY (idConversacion), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));	CREATE TABLE PARTICIPANTES(idParticipante SERIAL NOT NULL, idConversacion INT NOT NULL, idUsuario INT NOT NULL, tipoUsuario VARCHAR(255) NOT NULL CHECK (tipoUsuario IN ('USER', 'ADMIN', 'MODERADOR')), CONSTRAINT idParticipante_pk PRIMARY KEY (idParticipante), CONSTRAINT idConversacion_fk FOREIGN KEY (idConversacion) REFERENCES CONVERSACION (idConversacion), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));
CREATE TABLE MENSAJES(idMensaje SERIAL NOT NULL, idConversacion INT NOT NULL, idUsuario INT NOT NULL, tipoMensaje VARCHAR(10) CHECK (tipoMensaje IN ('TEXT', 'IMAGE', 'VIDEO')), mensaje VARCHAR(255) NOT NULL, fechEnvio TIMESTAMP DEFAULT CURRENT_TIMESTAMP, fechEliminacion TIMESTAMP, 	CREATE TABLE MENSAJES_ELIMINADOS(idMsj SERIAL NOT NULL, idMensaje INT NOT NULL, idUsuario INT NOT NULL, fechEliminado TIMESTAMP, CONSTRAINT idMsj_pk PRIMARY KEY (idMsj), CONSTRAINT idMensaje_fk FOREIGN KEY (idMensaje) REFERENCES MENSAJES (idMesaje), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));

CONSTRAINT idMensaje_pk PRIMARY KEY (idMensaje), CONSTRAINT idConversacion_fk FOREIGN KEY (idConversacion) REFERENCES CONVERSACION (idConversacion), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));	
CREATE TABLE CONVERSACION_ELIMINADA(idConv SERIAL NOT NULL, idConversacion INT NOT NULL, idUsuario INT NOT NULL, fechElimimacion TIMESTAMP, CONSTRAINT idConv_pk PRIMARY KEY (idConv), CONSTRAINT idConversacion_fk FOREIGN KEY (idConversacion) REFERENCES CONVERSACION (idConversacion), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));	CREATE TABLE ACCESO(idAcceso SERIAL NOT NULL, idUsuario INT NOT NULL, idDispositivo INT NOT NULL, fecRegistro TIMESTAMP DEFAULT CURRENT_TIMESTAMP, fecActualizacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP, CONSTRAINT idAcceso_pk PRIMARY KEY (idAcceso), CONSTRAINT idDispositivo_fk FOREIGN KEY (idDispositivo) REFERENCES DISPOSITIVOS (idDispositivo), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));
CREATE TABLE USUARIOS_BLOQUEADOS(idUsBloqueo SERIAL NOT NULL, idUsuario INT NOT NULL, idParticipante INT NOT NULL, telefono VARCHAR(255) NOT NULL, fechaBloqueo TIMESTAMP DEFAULT CURRENT_TIMESTAMP, CONSTRAINT idUsBloqueo_pk PRIMARY KEY (idUsBloqueo), CONSTRAINT idParticipante_fk FOREIGN KEY (idParticipante) REFERENCES PARTICIPANTES (idParticipante), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));	CREATE TABLE REPORTES(idReporte SERIAL NOT NULL, idUsuario INT NOT NULL, idParticipante INT NOT NULL, tipoReporte VARCHAR(255) NOT NULL, nota VARCHAR(255) NOT NULL, fechReporte TIMESTAMP DEFAULT CURRENT_TIMESTAMP, CONSTRAINT idReporte_pk PRIMARY KEY (idReporte), CONSTRAINT idParticipante_fk FOREIGN KEY (idParticipante) REFERENCES PARTICIPANTES (idParticipante), CONSTRAINT idUsuario_fk FOREIGN KEY (idUsuario) REFERENCES USUARIOS (idUsuario));

CREACION DE LA BASE DE DATOS EN POSTGRESQL Y pgADMIN4







public.usuarios/Db_Mensajeria/ServidorEduardo@CN_Bases

Sin límite

Consulta | Historial de Consultas

```
SELECT * FROM public.usuarios
ORDER BY idusuario ASC
```

Data Output | Mensajes | Notificaciones

	idusuario [PK] integer	nombre character varying (255)	apellido character varying (255)	username character varying (255)	password character varying (255)	email character varying (255)
1	1	Juan	Pérez	juanp	password1	juanp@example.com
2	2	María	López	marial	password2	marial@example.com
3	3	Carlos	Sánchez	carloss	password3	carloss@example.com
4	4	Ana	Martínez	anam	password4	anam@example.com
5	5	Luis	García	luisg	password5	luisg@example.com
6	6	Carmen	Gómez	carmeng	password6	carmeng@example.com
7	7	Pedro	Fernández	pedrof	password7	pedrof@example.com
8	8	Laura	Rodríguez	laurar	password8	laurar@example.com
9	9	Miguel	Hernández	miguelh	password9	miguelh@example.com
10	10	Sara	Díaz	sarad	password10	sarad@example.com
11	11	Andrés	Ruiz	andresr	password11	andresr@example.com
12	12	Elena	Jiménez	elenaj	password12	elenaj@example.com
13	13	Jorge	Moreno	jorgem	password13	jorgem@example.com
14	14	Patricia	Muñoz	patriciam	password14	patriciam@example.com
15	15	Ricardo	Romero	ricardor	password15	ricardor@example.com

Total rows: 20 of 20 | Query complete 00:00:00.479 | Ln 1, Col 1

public.reportes/Db_Mensajeria/ServidorEduardo@CN_Bases

Sin límite

Consulta | Historial de Consultas

```
SELECT * FROM public.reportes
ORDER BY idreporte ASC
```

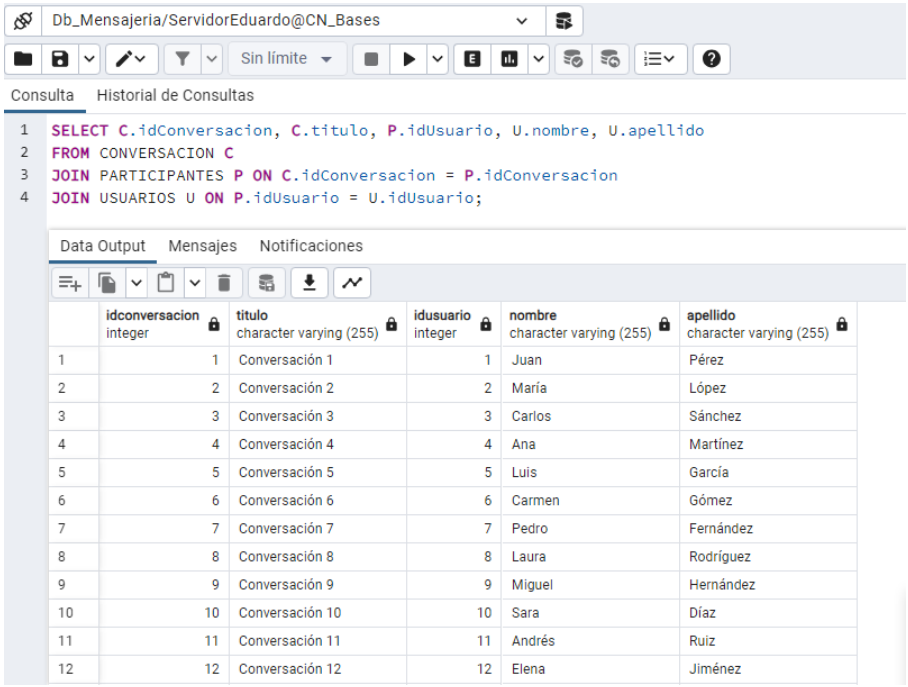
Data Output | Mensajes | Notificaciones

	idreporte [PK] integer	idusuario integer	idparticipante integer	tiporeporte character varying (255)	nota character varying (255)	fechreporte timestamp without time zone
1	1	1	1	Spam	Usuario envía spam.	2024-06-13 16:25:06.434539
2	2	2	2	Abuso	Usuario ha sido abusivo.	2024-06-13 16:25:06.434539
3	3	3	3	Contenido inapropiado	Usuario comparte contenido inapropiado.	2024-06-13 16:25:06.434539
4	4	4	4	Spam	Usuario envía spam.	2024-06-13 16:25:06.434539
5	5	5	5	Abuso	Usuario ha sido abusivo.	2024-06-13 16:25:06.434539
6	6	6	6	Contenido inapropiado	Usuario comparte contenido inapropiado.	2024-06-13 16:25:06.434539
7	7	7	7	Spam	Usuario envía spam.	2024-06-13 16:25:06.434539
8	8	8	8	Abuso	Usuario ha sido abusivo.	2024-06-13 16:25:06.434539
9	9	9	9	Contenido inapropiado	Usuario comparte contenido inapropiado.	2024-06-13 16:25:06.434539
10	10	10	10	Spam	Usuario envía spam.	2024-06-13 16:25:06.434539
11	11	11	11	Abuso	Usuario ha sido abusivo.	2024-06-13 16:25:06.434539
12	12	12	12	Contenido inapropiado	Usuario comparte contenido inapropiado.	2024-06-13 16:25:06.434539
13	13	13	13	Spam	Usuario envía spam.	2024-06-13 16:25:06.434539
14	14	14	14	Abuso	Usuario ha sido abusivo.	2024-06-13 16:25:06.434539
15	15	15	15	Contenido inapropiado	Usuario comparte contenido inapropiado.	2024-06-13 16:25:06.434539

Total rows: 20 of 20 | Query complete 00:00:00.454 | Ln 1, Col 1

CONSULTAS.

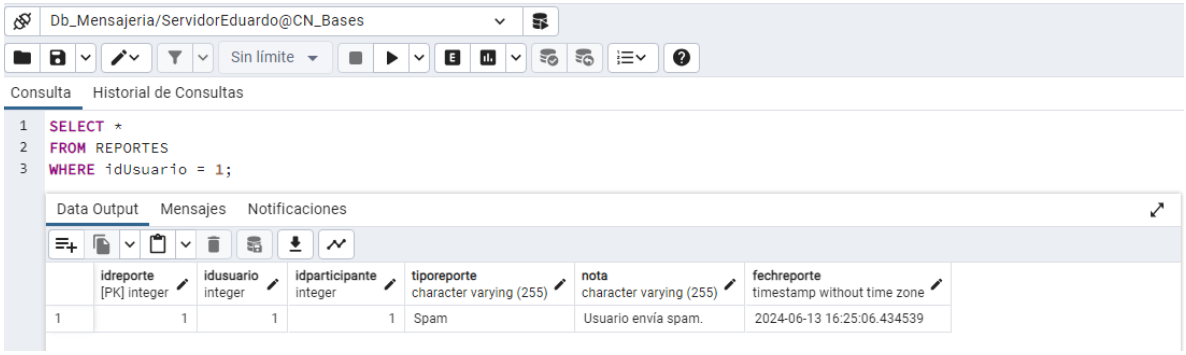
1. Obtener todas las conversaciones y sus respectivos participantes.



```
SELECT C.idConversacion, C.titulo, P.idUsuario, U.nombre, U.apellido
FROM CONVERSACION C
JOIN PARTICIPANTES P ON C.idConversacion = P.idConversacion
JOIN USUARIOS U ON P.idUsuario = U.idUsuario;
```

	idconversacion integer	titulo character varying (255)	idusuario integer	nombre character varying (255)	apellido character varying (255)
1	1	Conversación 1	1	Juan	Pérez
2	2	Conversación 2	2	María	López
3	3	Conversación 3	3	Carlos	Sánchez
4	4	Conversación 4	4	Ana	Martínez
5	5	Conversación 5	5	Luis	García
6	6	Conversación 6	6	Carmen	Gómez
7	7	Conversación 7	7	Pedro	Fernández
8	8	Conversación 8	8	Laura	Rodríguez
9	9	Conversación 9	9	Miguel	Hernández
10	10	Conversación 10	10	Sara	Díaz
11	11	Conversación 11	11	Andrés	Ruiz
12	12	Conversación 12	12	Elena	Jiménez

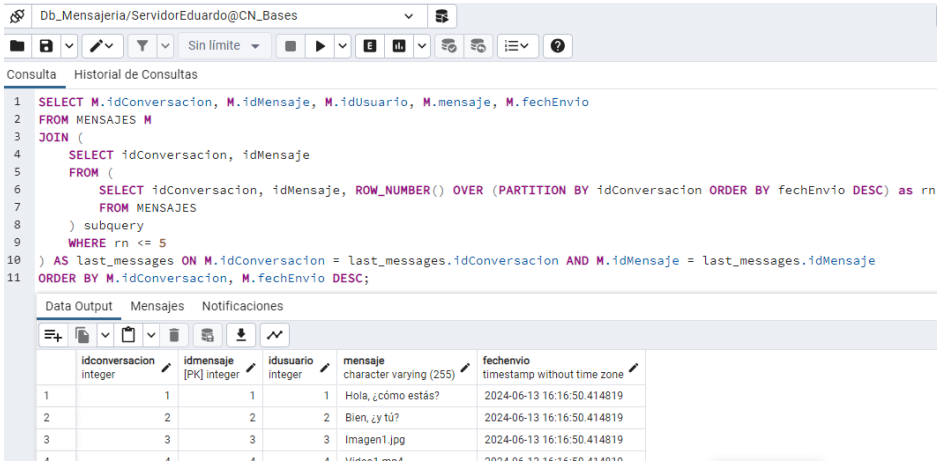
2. Obtener todos los reportes realizados por un usuario específico.



```
SELECT *
FROM REPORTES
WHERE idUsuario = 1;
```

	idreporte [PK] integer	idusuario integer	idparticipante integer	tiporeporte character varying (255)	nota character varying (255)	fechreporte timestamp without time zone
1	1	1	1	Spam	Usuario envía spam.	2024-06-13 16:25:06.434539

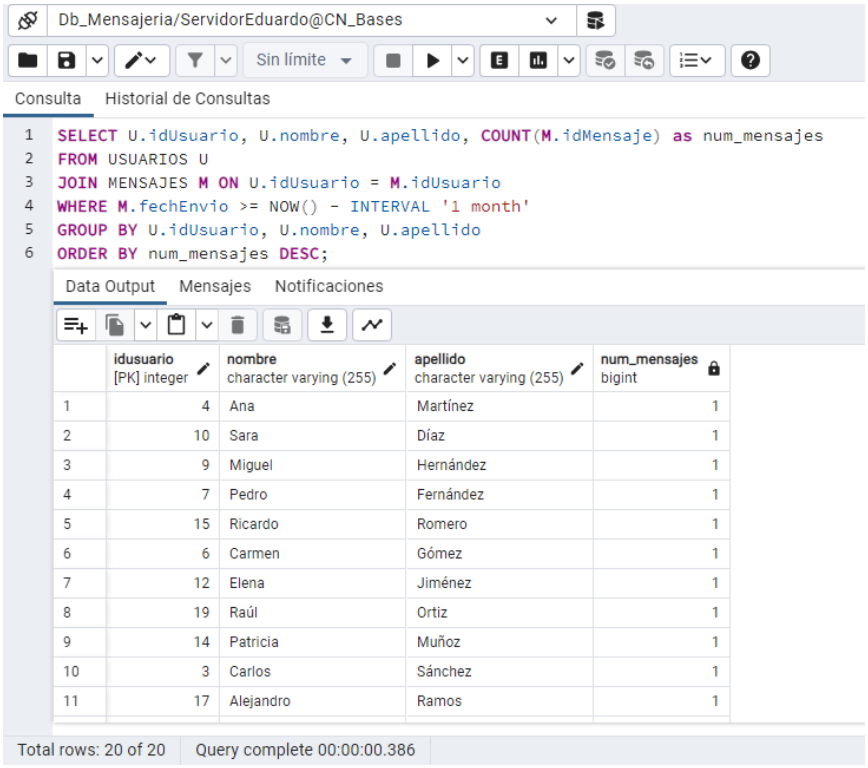
3. Obtener los últimos 5 mensajes enviados por cada conversación.



```
SELECT M.idConversacion, M.idMensaje, M.idUsuario, M.mensaje, M.fechEnvio
FROM MENSAJES M
JOIN (
    SELECT idConversacion, idMensaje
    FROM (
        SELECT idConversacion, idMensaje, ROW_NUMBER() OVER (PARTITION BY idConversacion ORDER BY fechEnvio DESC) as rn
        FROM MENSAJES
    ) subquery
    WHERE rn <= 5
) AS last_messages ON M.idConversacion = last_messages.idConversacion AND M.idMensaje = last_messages.idMensaje
ORDER BY M.idConversacion, M.fechEnvio DESC;
```

	idconversacion integer	idmensaje [PK] integer	idusuario integer	mensaje character varying (255)	fechenvio timestamp without time zone
1	1	1	1	Hola, ¿cómo estás?	2024-06-13 16:16:50.414819
2	2	2	2	Bien, ¿y tú?	2024-06-13 16:16:50.414819
3	3	3	3	Imagen1.jpg	2024-06-13 16:16:50.414819

4. Obtener la lista de conversaciones con más de 10 participantes.



Db_Mensajeria/ServidorEduardo@CN_Bases

Sin límite

Consulta | Historial de Consultas

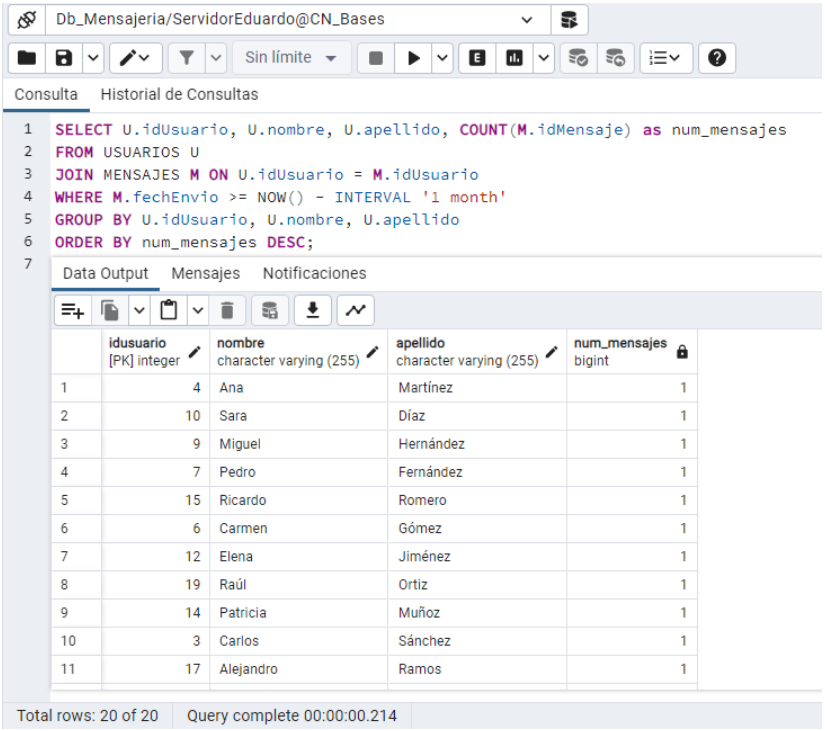
```
1  SELECT U.idUsuario, U.nombre, U.apellido, COUNT(M.idMensaje) as num_mensajes
2  FROM USUARIOS U
3  JOIN MENSAJES M ON U.idUsuario = M.idUsuario
4  WHERE M.fechEnvio >= NOW() - INTERVAL '1 month'
5  GROUP BY U.idUsuario, U.nombre, U.apellido
6  ORDER BY num_mensajes DESC;
```

Data Output | Mensajes | Notificaciones

	idusuario [PK] integer	nombre character varying (255)	apellido character varying (255)	num_mensajes bigint
1	4	Ana	Martínez	1
2	10	Sara	Díaz	1
3	9	Miguel	Hernández	1
4	7	Pedro	Fernández	1
5	15	Ricardo	Romero	1
6	6	Carmen	Gómez	1
7	12	Elena	Jiménez	1
8	19	Raúl	Ortiz	1
9	14	Patricia	Muñoz	1
10	3	Carlos	Sánchez	1
11	17	Alejandro	Ramos	1

Total rows: 20 of 20 | Query complete 00:00:00.386

5. Obtener el número de mensajes enviados por cada usuario en el último mes, ordenando por la cantidad de mensajes.



Db_Mensajeria/ServidorEduardo@CN_Bases

Sin límite

Consulta | Historial de Consultas

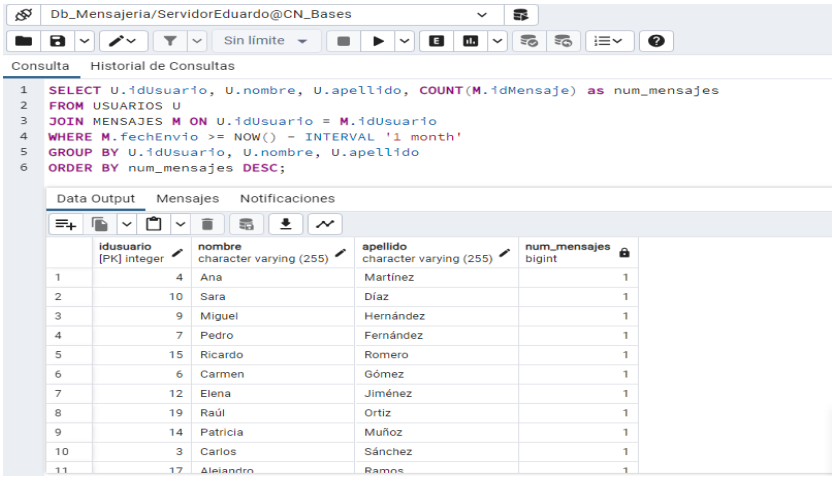
```
1  SELECT U.idUsuario, U.nombre, U.apellido, COUNT(M.idMensaje) as num_mensajes
2  FROM USUARIOS U
3  JOIN MENSAJES M ON U.idUsuario = M.idUsuario
4  WHERE M.fechEnvio >= NOW() - INTERVAL '1 month'
5  GROUP BY U.idUsuario, U.nombre, U.apellido
6  ORDER BY num_mensajes DESC;
7
```

Data Output | Mensajes | Notificaciones

	idusuario [PK] integer	nombre character varying (255)	apellido character varying (255)	num_mensajes bigint
1	4	Ana	Martínez	1
2	10	Sara	Díaz	1
3	9	Miguel	Hernández	1
4	7	Pedro	Fernández	1
5	15	Ricardo	Romero	1
6	6	Carmen	Gómez	1
7	12	Elena	Jiménez	1
8	19	Raúl	Ortiz	1
9	14	Patricia	Muñoz	1
10	3	Carlos	Sánchez	1
11	17	Alejandro	Ramos	1

Total rows: 20 of 20 | Query complete 00:00:00.214

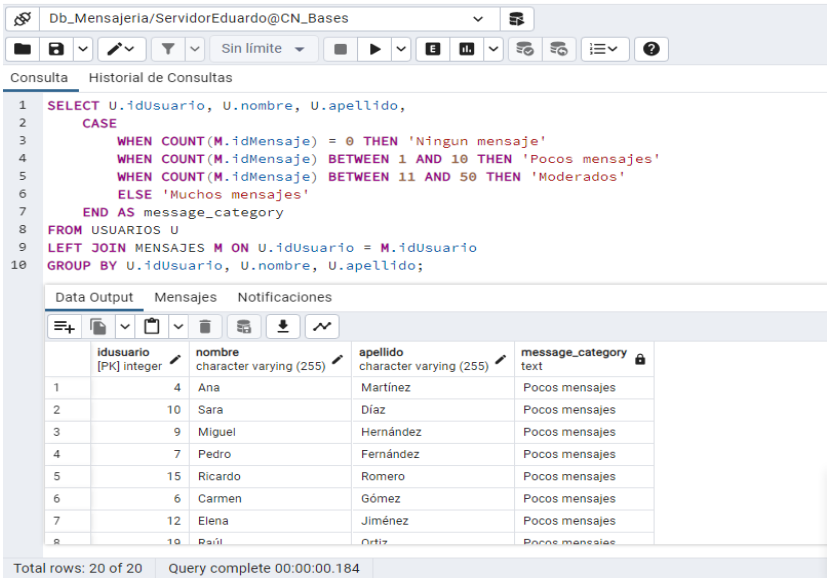
6. Obtener las conversaciones más activas, con más mensajes en la última semana.



```
SELECT U.idUsuario, U.nombre, U.apellido, COUNT(M.idMensaje) as num_mensajes
FROM USUARIOS U
JOIN MENSAJES M ON U.idUsuario = M.idUsuario
WHERE M.fechEnvio >= NOW() - INTERVAL '1 month'
GROUP BY U.idUsuario, U.nombre, U.apellido
ORDER BY num_mensajes DESC;
```

	idusuario [PK] integer	nombre character varying (255)	apellido character varying (255)	num_mensajes bigint
1	4	Ana	Martínez	1
2	10	Sara	Díaz	1
3	9	Miguel	Hernández	1
4	7	Pedro	Fernández	1
5	15	Ricardo	Romero	1
6	6	Carmen	Gómez	1
7	12	Elena	Jiménez	1
8	19	Raúl	Ortiz	1
9	14	Patricia	Muñoz	1
10	3	Carlos	Sánchez	1
11	17	Alejandro	Ramos	1

7. Usar case para categorizar usuarios según la cantidad de mensajes enviados.

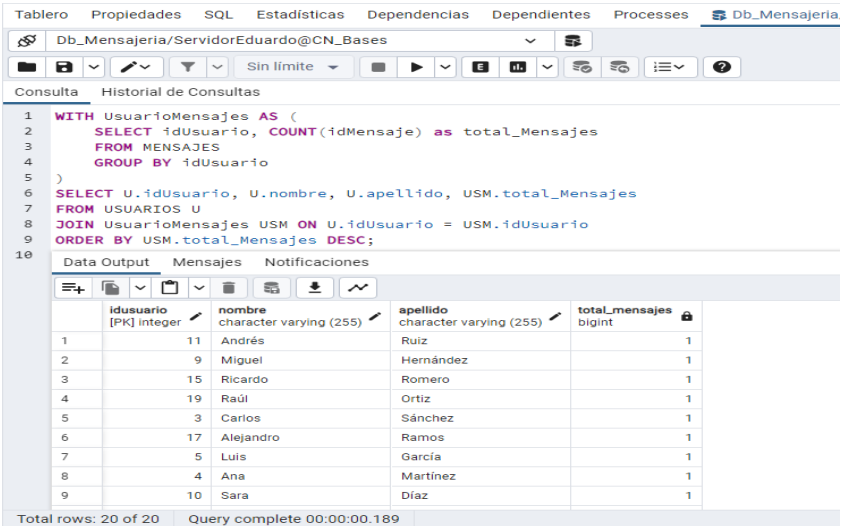


```
SELECT U.idUsuario, U.nombre, U.apellido,
    CASE
        WHEN COUNT(M.idMensaje) = 0 THEN 'Ningun mensaje'
        WHEN COUNT(M.idMensaje) BETWEEN 1 AND 10 THEN 'Pocos mensajes'
        WHEN COUNT(M.idMensaje) BETWEEN 11 AND 50 THEN 'Moderados'
        ELSE 'Muchos mensajes'
    END AS message_category
FROM USUARIOS U
LEFT JOIN MENSAJES M ON U.idUsuario = M.idUsuario
GROUP BY U.idUsuario, U.nombre, U.apellido;
```

	idusuario [PK] integer	nombre character varying (255)	apellido character varying (255)	message_category text
1	4	Ana	Martínez	Pocos mensajes
2	10	Sara	Díaz	Pocos mensajes
3	9	Miguel	Hernández	Pocos mensajes
4	7	Pedro	Fernández	Pocos mensajes
5	15	Ricardo	Romero	Pocos mensajes
6	6	Carmen	Gómez	Pocos mensajes
7	12	Elena	Jiménez	Pocos mensajes
8	19	Raúl	Ortiz	Pocos mensajes

Total rows: 20 of 20 Query complete 00:00:00.184

8. Usar CTEs para calcular el número total de mensajes enviados por cada usuario y luego obtener los usuarios con más mensajes.

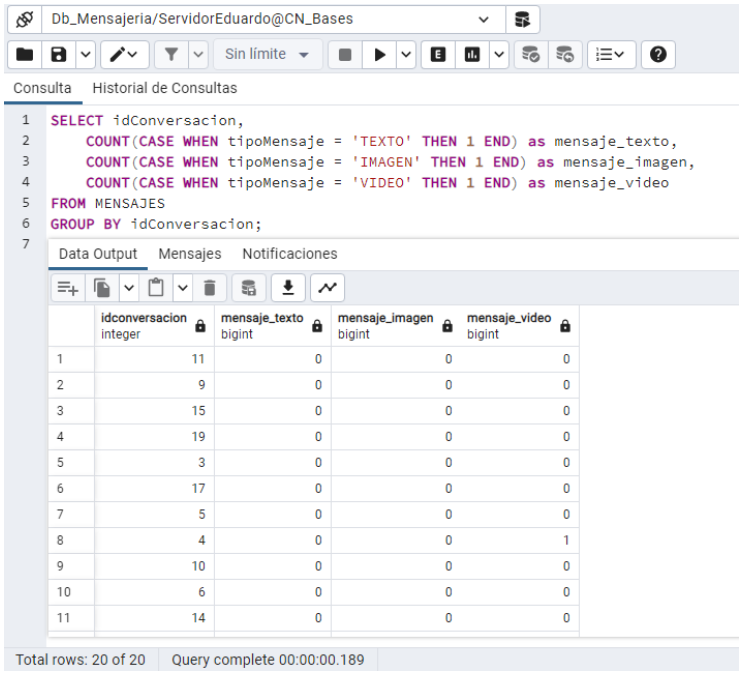


```
WITH UsuarioMensajes AS (
    SELECT idUsuario, COUNT(idMensaje) as total_Mensajes
    FROM MENSAJES
    GROUP BY idUsuario
)
SELECT U.idUsuario, U.nombre, U.apellido, USM.total_Mensajes
FROM USUARIOS U
JOIN UsuarioMensajes USM ON U.idUsuario = USM.idUsuario
ORDER BY USM.total_Mensajes DESC;
```

	idusuario [PK] integer	nombre character varying (255)	apellido character varying (255)	total_mensajes bigint
1	11	Andrés	Ruiz	1
2	9	Miguel	Hernández	1
3	15	Ricardo	Romero	1
4	19	Raúl	Ortiz	1
5	3	Carlos	Sánchez	1
6	17	Alejandro	Ramos	1
7	5	Luis	García	1
8	4	Ana	Martínez	1
9	10	Sara	Díaz	1

Total rows: 20 of 20 Query complete 00:00:00.189

9. Usar case dentro de una función de agregación para contar mensajes de diferentes tipos.



Db_Mensajeria/ServidorEduardo@CN_Bases

Sin límite

Consulta | Historial de Consultas

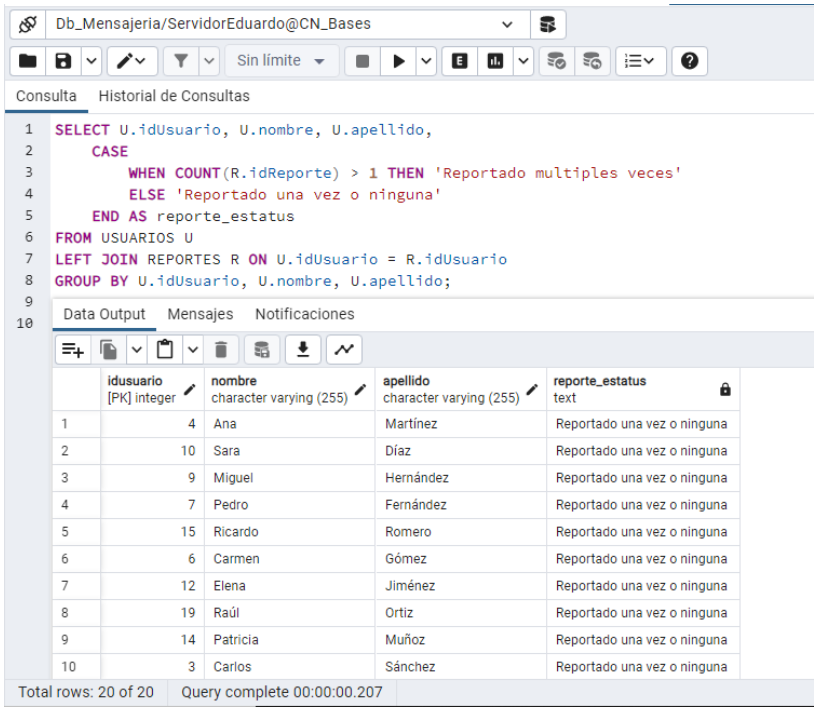
```
SELECT idConversacion,
    COUNT(CASE WHEN tipoMensaje = 'TEXTO' THEN 1 END) as mensaje_texto,
    COUNT(CASE WHEN tipoMensaje = 'IMAGEN' THEN 1 END) as mensaje_imagen,
    COUNT(CASE WHEN tipoMensaje = 'VIDEO' THEN 1 END) as mensaje_video
FROM MENSAJES
GROUP BY idConversacion;
```

Data Output | Mensajes | Notificaciones

	idconversacion integer	mensaje_texto bigint	mensaje_imagen bigint	mensaje_video bigint
1	11	0	0	0
2	9	0	0	0
3	15	0	0	0
4	19	0	0	0
5	3	0	0	0
6	17	0	0	0
7	5	0	0	0
8	4	0	0	1
9	10	0	0	0
10	6	0	0	0
11	14	0	0	0

Total rows: 20 of 20 | Query complete 00:00:00.189

10. Usar case para crear una columna calculada que indique si in usuario ha sido reportado más de una vez.



Db_Mensajeria/ServidorEduardo@CN_Bases

Sin límite

Consulta | Historial de Consultas

```
SELECT U.idUsuario, U.nombre, U.apellido,
    CASE
        WHEN COUNT(R.idReporte) > 1 THEN 'Reportado multiples veces'
        ELSE 'Reportado una vez o ninguna'
    END AS reporte_estatus
FROM USUARIOS U
LEFT JOIN REPORTES R ON U.idUsuario = R.idUsuario
GROUP BY U.idUsuario, U.nombre, U.apellido;
```

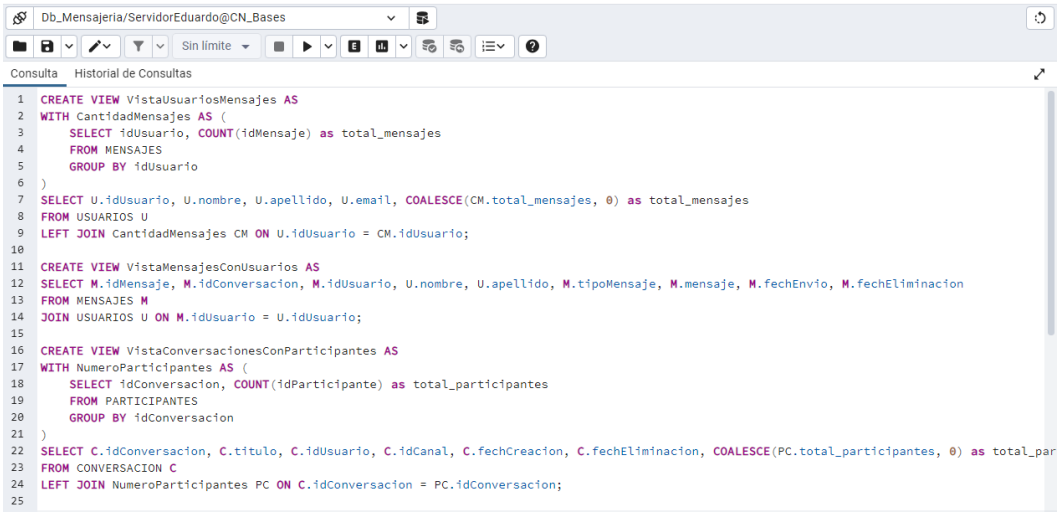
Data Output | Mensajes | Notificaciones

	idusuario [PK] integer	nombre character varying (255)	apellido character varying (255)	reporte_estatus text
1	4	Ana	Martínez	Reportado una vez o ninguna
2	10	Sara	Díaz	Reportado una vez o ninguna
3	9	Miguel	Hernández	Reportado una vez o ninguna
4	7	Pedro	Fernández	Reportado una vez o ninguna
5	15	Ricardo	Romero	Reportado una vez o ninguna
6	6	Carmen	Gómez	Reportado una vez o ninguna
7	12	Elena	Jiménez	Reportado una vez o ninguna
8	19	Raúl	Ortiz	Reportado una vez o ninguna
9	14	Patricia	Muñoz	Reportado una vez o ninguna
10	3	Carlos	Sánchez	Reportado una vez o ninguna

Total rows: 20 of 20 | Query complete 00:00:00.207

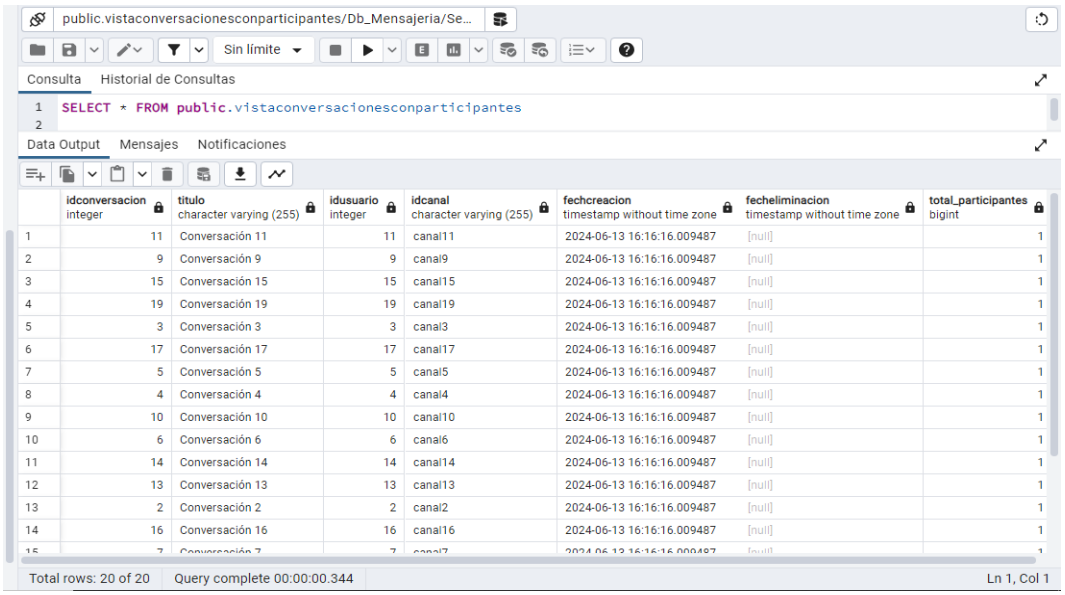
Nota: CTEs (Common Table Expressions) son una característica de SQL que permite definir expresiones de tabla temporales en la cual se puede utilizar en una consulta principal. Estos se definen con la cláusula WITH y pueden mejorar la legibilidad y la organización de las consultas SQL.

VISTAS




```sql
CREATE VIEW VistaUsuariosMensajes AS
WITH CantidadMensajes AS (
    SELECT idUsuario, COUNT(idMensaje) as total_mensajes
    FROM MENSAJES
    GROUP BY idUsuario
)
SELECT U.idUsuario, U.nombre, U.apellido, U.email, COALESCE(CM.total_mensajes, 0) as total_mensajes
FROM USUARIOS U
LEFT JOIN CantidadMensajes CM ON U.idUsuario = CM.idUsuario;

CREATE VIEW VistaMensajesConUsuarios AS
SELECT M.idMensaje, M.idConversacion, M.idUsuario, U.nombre, U.apellido, M.tipoMensaje, M.mensaje, M.fechEnvio, M.fechEliminacion
FROM MENSAJES M
JOIN USUARIOS U ON M.idUsuario = U.idUsuario;

CREATE VIEW VistaConversacionesConParticipantes AS
WITH NumeroParticipantes AS (
    SELECT idConversacion, COUNT(idParticipante) as total_participantes
    FROM PARTICIPANTES
    GROUP BY idConversacion
)
SELECT C.idConversacion, C.titulo, C.idUsuario, C.idCanal, C.fechCreacion, C.fechEliminacion, COALESCE(PC.total_participantes, 0) as total_par
FROM CONVERSACION C
LEFT JOIN NumeroParticipantes PC ON C.idConversacion = PC.idConversacion;
```

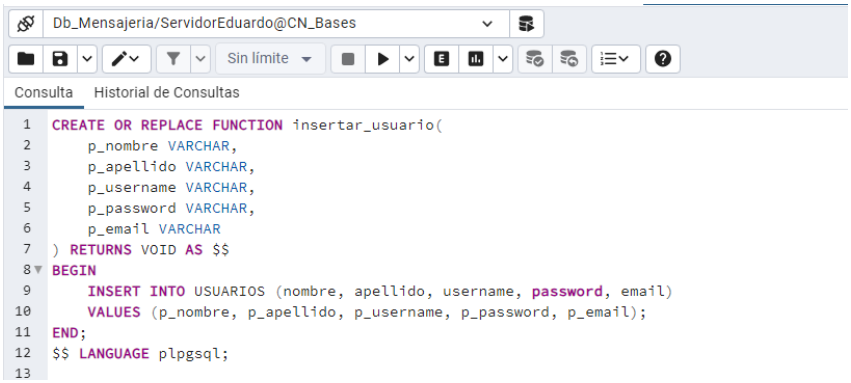
```sql

CREATE VIEW VistaUsuariosBloqueados AS
SELECT UB.idUsBloqueo, UB.idUsuario, U.nombre AS nombre_usuario, U.apellido AS apellido_usuario,
       UB.idParticipante, P.idUsuario AS idUsuario_bloqueado, UB.telefono, UB.fechaBloqueo
FROM USUARIOS_BLOQUEADOS UB
JOIN USUARIOS U ON UB.idUsuario = U.idUsuario
JOIN PARTICIPANTES P ON UB.idParticipante = P.idParticipante;

CREATE VIEW VistaReportes AS
SELECT R.idReporte, R.idUsuario, U.nombre AS nombre_reportador, U.apellido AS apellido_reportador,
       R.idParticipante, P.idUsuario AS idUsuario_reportado, R.tipoReporte, R.nota, R.fechReporte
FROM REPORTES R
JOIN USUARIOS U ON R.idUsuario = U.idUsuario
JOIN PARTICIPANTES P ON R.idParticipante = P.idParticipante;

```



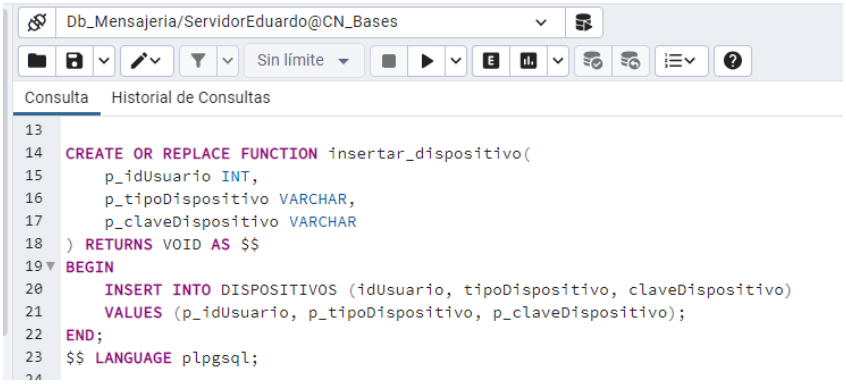
	idconversacion integer	titulo character varying (255)	idusuario integer	idcanal character varying (255)	fechcreacion timestamp without time zone	fecheliminacion timestamp without time zone	total_participantes bigint
1	11	Conversación 11	11	canal11	2024-06-13 16:16:16.009487	[null]	1
2	9	Conversación 9	9	canal9	2024-06-13 16:16:16.009487	[null]	1
3	15	Conversación 15	15	canal15	2024-06-13 16:16:16.009487	[null]	1
4	19	Conversación 19	19	canal19	2024-06-13 16:16:16.009487	[null]	1
5	3	Conversación 3	3	canal3	2024-06-13 16:16:16.009487	[null]	1
6	17	Conversación 17	17	canal17	2024-06-13 16:16:16.009487	[null]	1
7	5	Conversación 5	5	canal5	2024-06-13 16:16:16.009487	[null]	1
8	4	Conversación 4	4	canal4	2024-06-13 16:16:16.009487	[null]	1
9	10	Conversación 10	10	canal10	2024-06-13 16:16:16.009487	[null]	1
10	6	Conversación 6	6	canal6	2024-06-13 16:16:16.009487	[null]	1
11	14	Conversación 14	14	canal14	2024-06-13 16:16:16.009487	[null]	1
12	13	Conversación 13	13	canal13	2024-06-13 16:16:16.009487	[null]	1
13	2	Conversación 2	2	canal2	2024-06-13 16:16:16.009487	[null]	1
14	16	Conversación 16	16	canal16	2024-06-13 16:16:16.009487	[null]	1
15	7	Conversación 7	7	canal7	2024-06-13 16:16:16.009487	[null]	1

FUNCIONES (FUNCTION)

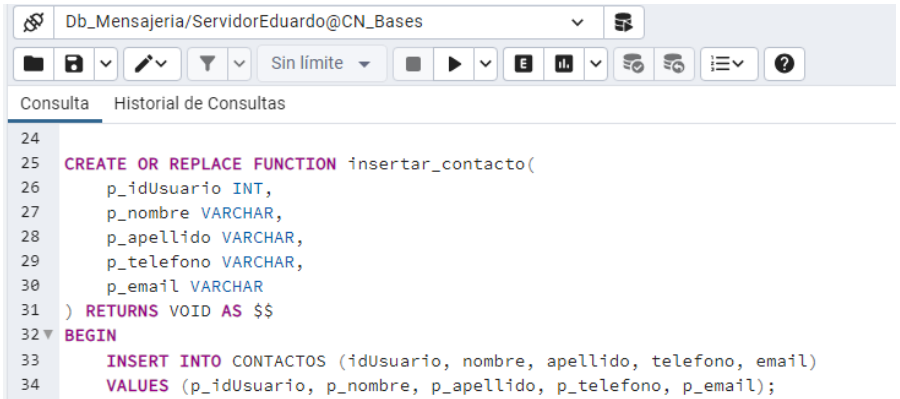
Se crearon algunas funciones, solo para algunas tablas que componen la base de datos.




```sql
CREATE OR REPLACE FUNCTION insertar_usuario(
    p_nombre VARCHAR,
    p_apellido VARCHAR,
    p_username VARCHAR,
    p_password VARCHAR,
    p_email VARCHAR
) RETURNS VOID AS $$
BEGIN
    INSERT INTO USUARIOS (nombre, apellido, username, password, email)
    VALUES (p_nombre, p_apellido, p_username, p_password, p_email);
END;
$$ LANGUAGE plpgsql;
```



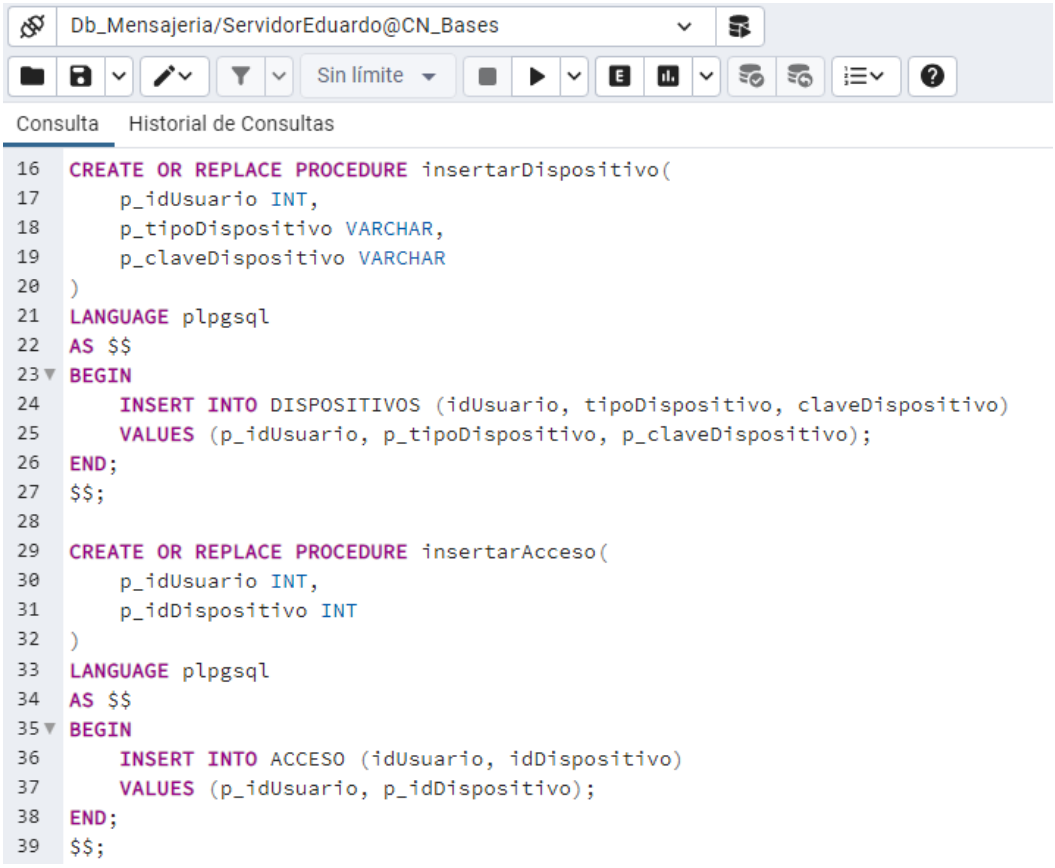

```sql

CREATE OR REPLACE FUNCTION insertar_dispositivo(
    p_idUsuario INT,
    p_tipoDispositivo VARCHAR,
    p_claveDispositivo VARCHAR
) RETURNS VOID AS $$
BEGIN
    INSERT INTO DISPOSITIVOS (idUsuario, tipoDispositivo, claveDispositivo)
    VALUES (p_idUsuario, p_tipoDispositivo, p_claveDispositivo);
END;
$$ LANGUAGE plpgsql;
```



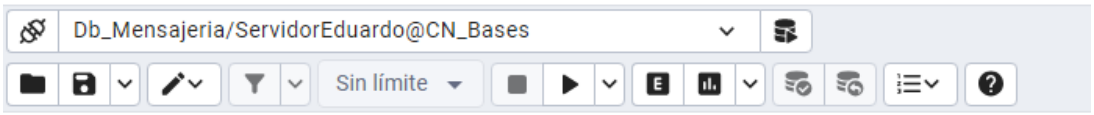

```sql

CREATE OR REPLACE FUNCTION insertar_contacto(
    p_idUsuario INT,
    p_nombre VARCHAR,
    p_apellido VARCHAR,
    p_telefono VARCHAR,
    p_email VARCHAR
) RETURNS VOID AS $$
BEGIN
    INSERT INTO CONTACTOS (idUsuario, nombre, apellido, telefono, email)
    VALUES (p_idUsuario, p_nombre, p_apellido, p_telefono, p_email);
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION insertar_acceso(
    p_idUsuario INT,
    p_idDispositivo INT
) RETURNS VOID AS $$
BEGIN
    INSERT INTO ACCESO (idUsuario, idDispositivo)
    VALUES (p_idUsuario, p_idDispositivo);
END;
$$ LANGUAGE plpgsql;
```

PROCEDIMIENTOS ALMACENADOS

Se crearon al igual que las funciones solo un par de procedimientos, para hacer pruebas en la base de datos.

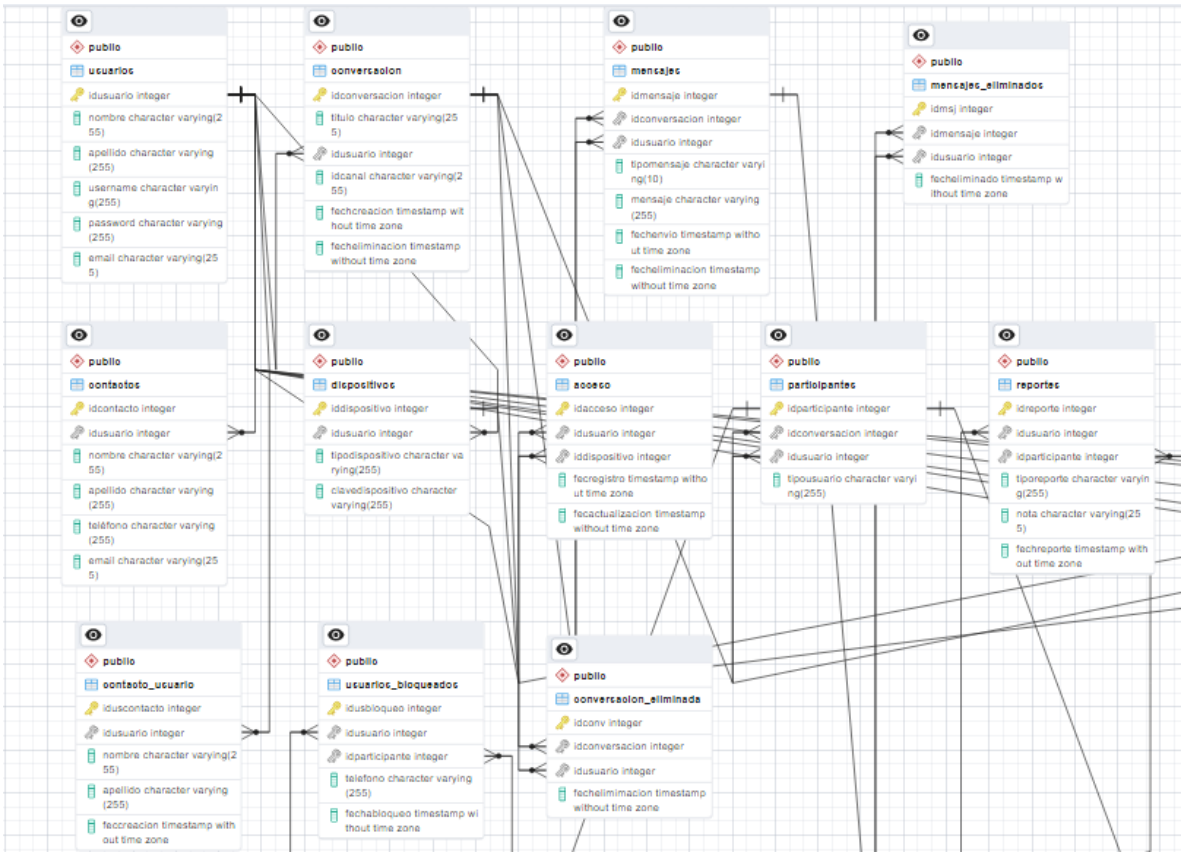



```sql
CREATE OR REPLACE PROCEDURE insertarDispositivo(
    p_idUsuario INT,
    p_tipoDispositivo VARCHAR,
    p_claveDispositivo VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO DISPOSITIVOS (idUsuario, tipoDispositivo, claveDispositivo)
    VALUES (p_idUsuario, p_tipoDispositivo, p_claveDispositivo);
END;
$$;

CREATE OR REPLACE PROCEDURE insertarAcceso(
    p_idUsuario INT,
    p_idDispositivo INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO ACCESO (idUsuario, idDispositivo)
    VALUES (p_idUsuario, p_idDispositivo);
END;
$$;
```




```sql
CREATE OR REPLACE PROCEDURE insertarUsuario(
    p_nombre VARCHAR,
    p_apellido VARCHAR,
    p_username VARCHAR,
    p_password VARCHAR,
    p_email VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO USUARIOS (nombre, apellido, username, password, email)
    VALUES (p_nombre, p_apellido, p_username, p_password, p_email);
END;
$$;

```

MODELO ER DE LA BASE DE DATOS.



Esta creación de base de datos fue construida a partir de todos los conocimientos adquiridos en la carrera de Ingeniería en Software, especialmente por los cursos de Bases de Datos, Especialidad de bases de datos 1, Especialidad de bases de datos 2, Inteligencia de negocios y Análisis de Requisitos. De igual manera estos conocimientos se reforzaron en cursos, tutoriales y libros para los diferentes gestores como Oracle, MySQL, Postgres, SQL Server, aplicando SQL y PL/SQL, e algunos casos.

Creado por: Eduardo Soto.