UNIVERSIDAD TECNOLÓGICA
DE XICOTEPEC DE JUÁREZ

Documentación de BD
“Comité de trasplantes”

NOMBRE DE EQUIPO: KICS

INTEGRANTES:

Karen Alyn Fosado Rodríguez – 210764

Carlos Martin Hernández de Jesús -210496

Sebastián Márquez García -210505

ASIGNATURA: Administración de Base De datos

DOCENTE TITULAR: Marco A. Ramírez Hernández

PROGRAMA EDUCATIVO: Ing. en Desarrollo y Gestión de Software

PERIODO: Enero – Abril 2024

Contenido

BASE DE DATOS SQL

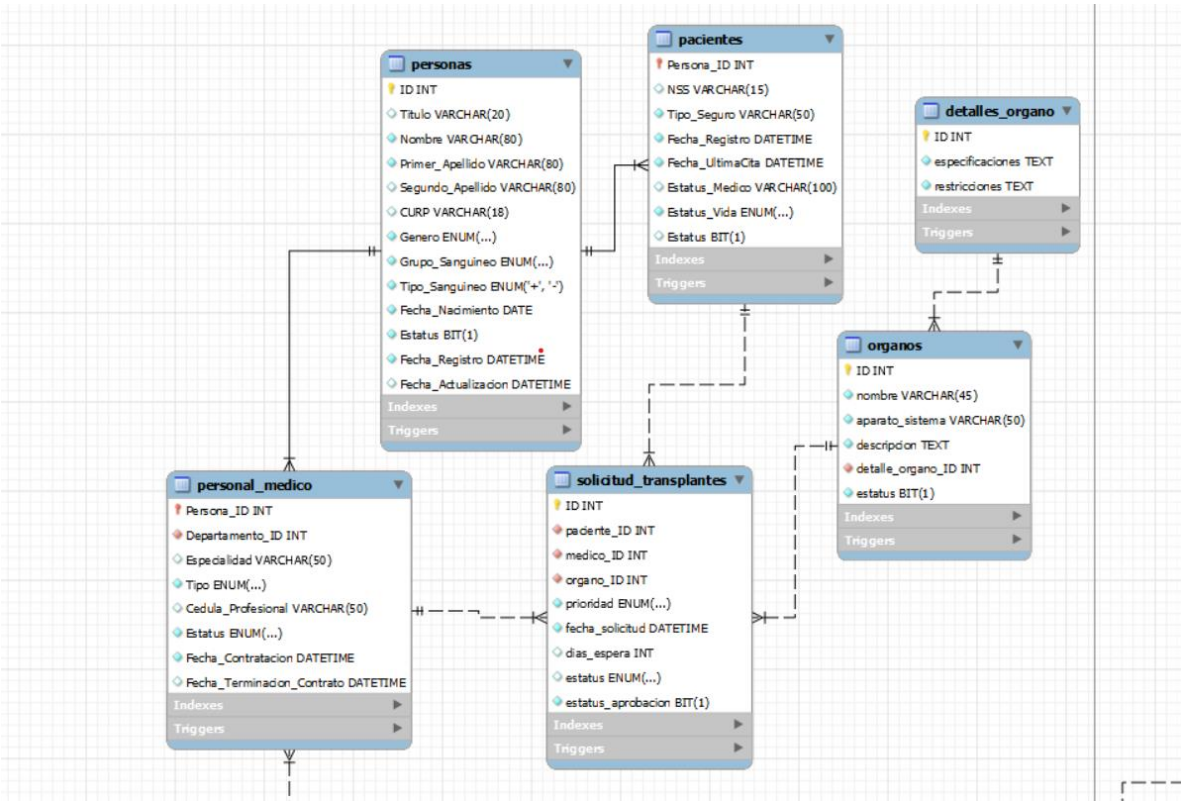
Descripción General

La base de datos "Hospital" fue diseñada para gestionar la información relacionada con pacientes, médicos, solicitudes de trasplantes, donadores, etc. Su objetivo principal es facilitar el seguimiento de los pacientes que han recibido trasplantes de órganos y proporcionar a los médicos y personal médico acceso a información relevante para el tratamiento y seguimiento de los pacientes.

La base de datos incluye tablas para almacenar información detallada sobre los pacientes, médicos, órganos disponibles para trasplante, así como las solicitudes de trasplantes realizadas y el estado de cada solicitud. También se registran datos sobre los donadores de órganos y su compatibilidad con los receptores.

El diseño de la base de datos se basa en un modelo entidad-relación (ER) que refleja las relaciones entre las diferentes entidades y sus atributos. Se han implementado restricciones de integridad referencial para garantizar la coherencia de los datos y se han definido procedimientos almacenados para automatizar tareas comunes y aplicar reglas de negocio.

Modelo de Datos



Diccionario de Datos

file:///D:/BD_PROYECTO_HOSPITAL/Diccionario%20de%20Datos%20Hospital%20-%20comite%20de%20transplantes.html

Reglas de negocio

1. Confidencialidad de la información: Todos los datos médicos y personales de los pacientes y donantes deben ser tratados con la máxima confidencialidad, de acuerdo con las regulaciones de protección de datos de salud.

2. Consentimiento informado: Antes de incluir a un paciente en la lista de espera o considerar a un donante para un trasplante, se debe obtener un consentimiento informado por escrito que explique los riesgos, beneficios y alternativas disponibles.

3. Priorización de trasplantes: Los trasplantes deben asignarse de manera justa y equitativa, priorizando a los pacientes en función de criterios médicos como la gravedad de la enfermedad, la compatibilidad del órgano y el tiempo en la lista de espera.

4. Actualización de la lista de espera: La lista de espera de trasplantes debe actualizarse regularmente con información precisa y actualizada sobre el estado de los pacientes y la disponibilidad de órganos.

5. Registro completo de donantes: Se debe mantener un registro completo y preciso de todos los donantes potenciales, incluyendo información biomédica, resultados de pruebas de compatibilidad y cualquier restricción o limitación relevante.

6. Transparencia en el proceso de asignación: El proceso de asignación de órganos debe ser transparente y estar sujeto a revisión, con registros detallados que documenten las decisiones tomadas y los criterios utilizados.

7. Seguimiento post-trasplante: Todos los pacientes trasplantados deben recibir un seguimiento continuo y coordinado para monitorear su progreso médico, detectar posibles complicaciones y garantizar una atención integral.

8. Coordinación interdisciplinaria: El comité de trasplantes debe colaborar estrechamente con otros departamentos y especialidades médicas, como cirugía, nefrología, hepatología y cuidados intensivos, para garantizar una atención integral y coordinada para los pacientes.

9. Gestión eficiente del inventario de órganos: El inventario de órganos disponibles para trasplante debe gestionarse de manera eficiente, asegurando una distribución equitativa y minimizando el tiempo de espera para los receptores.

10. Notificación oportuna de eventos críticos: Se deben establecer mecanismos de notificación rápida y eficaz para alertar al equipo médico sobre la disponibilidad de órganos compatibles,

cambios en el estado de los pacientes o cualquier evento crítico relacionado con los trasplantes.

11. Cumplimiento normativo y regulatorio: Todas las actividades relacionadas con el programa de trasplantes deben cumplir con las regulaciones y normativas locales, estatales y federales, así como con las pautas éticas y de práctica clínica establecidas.

12. Formación y capacitación del personal: El personal involucrado en el proceso de trasplante, incluidos los médicos, enfermeras y coordinadores de trasplantes, debe recibir una formación adecuada y actualizada sobre los procedimientos y protocolos establecidos.

Procedimientos almacenados y funciones:

- **Procesos Almacenados: sp_Insertar_Solicitudes_Transplante**

Descripción:

Este procedimiento almacenado se utiliza para insertar múltiples solicitudes de trasplante en la tabla solicitud_transplantes. Genera datos aleatorios para el paciente, médico, órgano, prioridad, fecha de solicitud y estado de la solicitud.

Parámetros:

cantidad: Número de solicitudes de trasplante que se desea insertar.

Funcionamiento:

1. Inicializa un contador v_i en 0.
2. En un bucle WHILE, genera datos aleatorios para cada solicitud de trasplante y la inserta en la tabla solicitud_transplantes.
3. Para cada solicitud:
    - Selecciona aleatoriamente un paciente, médico y órgano de las tablas correspondientes.
    - Genera una prioridad aleatoria (v_prioridad) entre 'urgente', 'alta' y 'moderada'.
    - Genera una fecha de solicitud aleatoria (v_fecha_solicitud) entre la fecha actual y hasta 200 días atrás.
    - Inserta la solicitud en la tabla solicitud_transplantes.

Ejemplo de uso:

call sp_Insertar_Solicitudes_Transplante(10);

Este comando insertará 10 solicitudes de trasplante en la tabla solicitud_transplantes.

- **Función: fn_calcular_dias_espera**

Descripción:

Esta función calcula los días de espera entre la fecha de solicitud de trasplante y la fecha actual. Toma la fecha de solicitud como argumento y devuelve el número de días de espera como un entero.

Parámetros:

fecha_solicitud: La fecha en la que se realizó la solicitud de trasplante.

Funcionamiento:

1. Inicializa una variable dias_espera para almacenar el resultado del cálculo.
2. Calcula la diferencia en días entre la fecha actual (NOW()) y la fecha de solicitud utilizando la función DATEDIFF().
3. Retorna el número de días de espera.

Ejemplo de Uso:

Esta función es mandada a llamar en un trigger, Este trigger se ejecuta antes de insertar una nueva fila en la tabla solicitud_transplantes. Su objetivo es calcular automáticamente los días de espera para un trasplante utilizando la función fn_calcular_dias_espera.

- Cuando se inserta una nueva fila en la tabla solicitud_transplantes, este trigger se activa automáticamente.
- Utiliza la función fn_calcular_dias_espera para calcular los días de espera basados en la fecha de solicitud de trasplante.
- Asigna el resultado del cálculo a la columna dias_espera de la nueva fila que se está insertando.

Ejemplo de uso:

Ejemplo de inserción en la tabla solicitud_transplantes ()

INSERT INTO solicitud_transplantes (paciente_ID, medico_ID, organo_ID, prioridad, fecha_solicitud, estatus, estatus_aprobacion)

VALUES (1, 2, 3, 'urgente', '2024-04-18 10:00:00', 'Transplante exitoso', 1);

Este comando insertará una nueva fila en la tabla solicitud_transplantes y automáticamente calculará los días de espera utilizando el trigger solicitud_transplantes_BEFORE_INSERT, de igual manera puede ser insertado con el proceso almacenado “sp_Insertar_Solicitudes_Transplante(10);”.

Vistas

- **Vista: vista_estatus_solicitudes**

Descripción

Esta vista proporciona un resumen del estado de las solicitudes de trasplantes en la tabla solicitud_transplantes, mostrando la cantidad de solicitudes en cada estado.

Definición

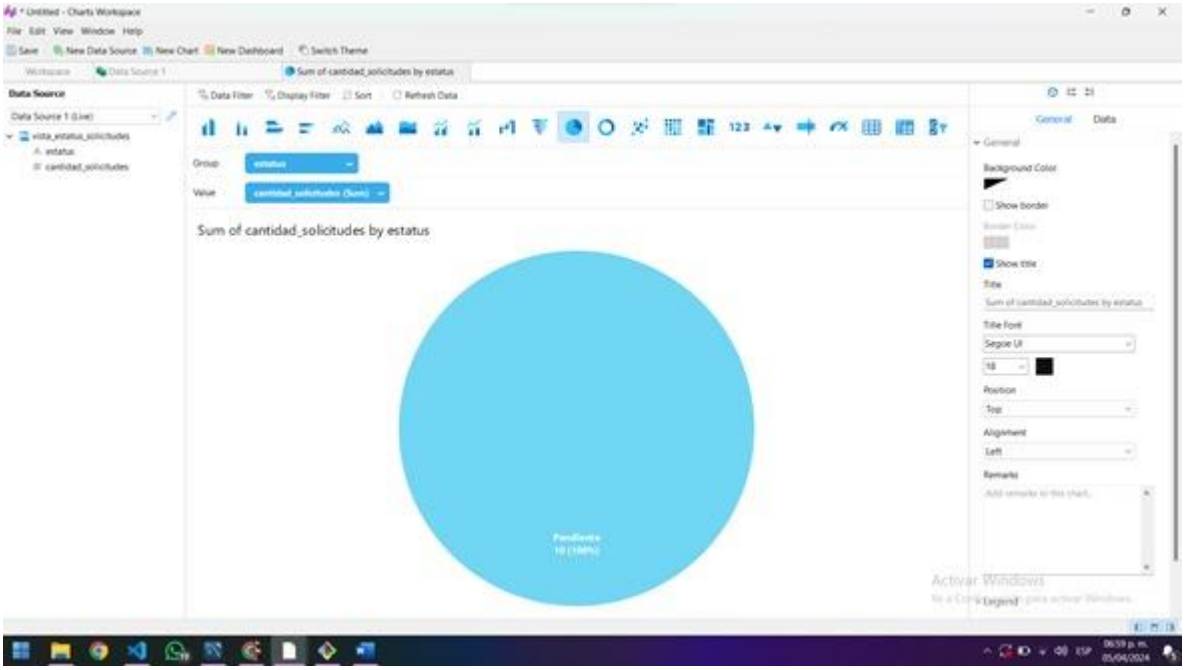
```
CREATE
  ALGORITHM = UNDEFINED
  DEFINER = `root`@`localhost`
  SQL SECURITY DEFINER
VIEW `vista_estatus_solicitudes` AS
  SELECT
    `solicitud_transplantes`.`estatus` AS `estatus`,
    COUNT(0) AS `cantidad_solicitudes`
  FROM
    `solicitud_transplantes`
  GROUP BY `solicitud_transplantes`.`estatus`;
```

Campos

- estatus: El estado de la solicitud de trasplante.
- cantidad_solicitudes: El número total de solicitudes en cada estado.

Uso

Esta vista puede ser consultada para obtener información sobre la distribución de las solicitudes de trasplantes según su estado. Por ejemplo (prueba de lectura de datos):



- **Vista: vista_distribucion_tipo_organo**

Descripción:

Esta vista muestra la distribución de las solicitudes de trasplantes según el tipo de órgano solicitado en la tabla solicitud_transplantes, contando el total de solicitudes para cada tipo de órgano.

Definición:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `vista_distribucion_tipo_organo` AS
    SELECT
        `o`.`nombre` AS `Tipo_Organo`,
        COUNT(0) AS `Total_Solicitudes`
    FROM
        (`solicitud_transplantes` `s`
```

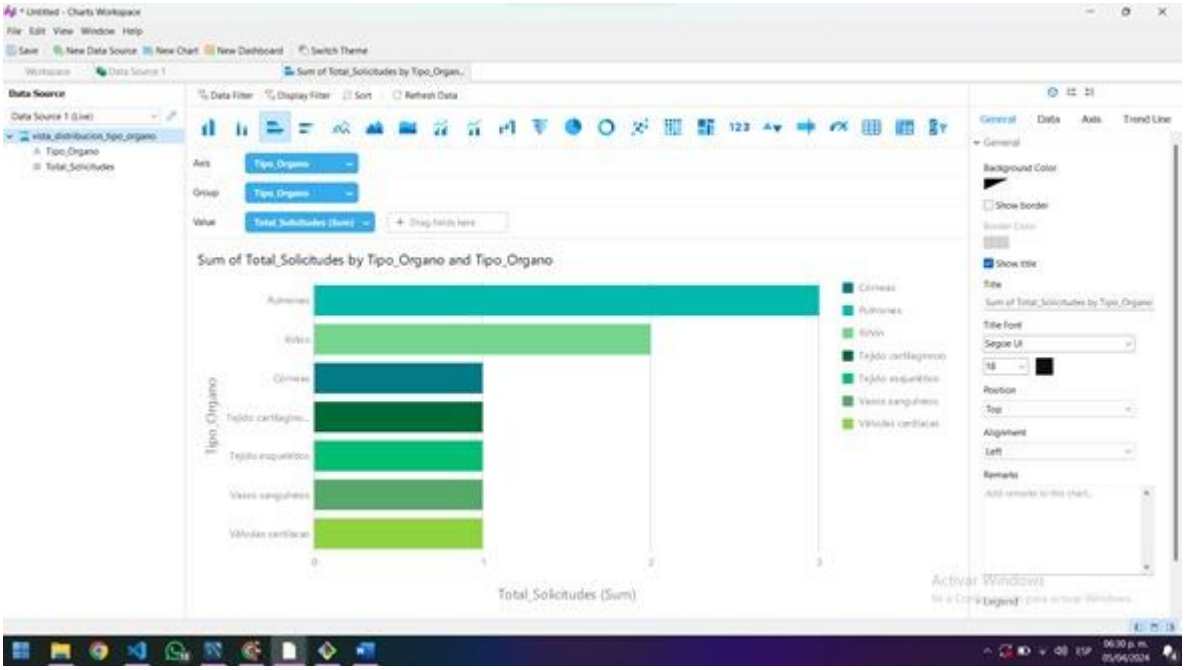
```
    JOIN `organos` `o` ON ((`s`.`organo_ID` = `o`.`ID`)))
GROUP BY `o`.`nombre`;
```

Campos

- Tipo_Organo: El tipo de órgano solicitado en la solicitud de trasplante.
- Total_Solicitudes: El número total de solicitudes para cada tipo de órgano.

Uso

Esta vista puede ser consultada para obtener información sobre la distribución de las solicitudes de trasplantes según el tipo de órgano solicitado. Por ejemplo (prueba de lectura de datos):



Usuarios, roles y privilegios:

```
/*Crear usuarios prueba  */

CREATE USER 'medico'@'localhost' IDENTIFIED BY 'contraseña';

CREATE USER 'paciente'@'localhost' IDENTIFIED BY 'contraseña';


/* Crear roles */

CREATE ROLE 'medico_role';

CREATE ROLE 'paciente_role';
```

```
/* Asignar privilegios a los roles */

GRANT SELECT, INSERT, UPDATE, DELETE ON bd_hospital_210764.* TO 'medico_role';

GRANT SELECT ON bd_hospital_210764.* TO 'paciente_role';


/* *Asignar roles a los usuarios:*/

GRANT medico_role TO 'medico'@'localhost';

GRANT paciente_role TO 'paciente'@'localhost';
```

Procesos de respaldo y recuperación

0 0 */15 * * mysqldump -u [usuario] -p[contraseña] [nombre_basedatos] > /ruta/para/guardar/respaldo.sql

En este ejemplo:

- 0 0 */15 * * especifica que la tarea se ejecute a la medianoche (00:00) de cada 15 días (el día exacto se determina automáticamente).
- mysqldump es el comando para realizar el respaldo.
- -u [usuario] especifica el nombre de usuario de la base de datos.
- -p[contraseña] especifica la contraseña del usuario (sin espacio entre -p y la contraseña).
- [nombre_basedatos] es el nombre de la base de datos que deseas respaldar
- > /ruta/para/guardar/respaldo.sql redirige la salida del comando mysqldump a un archivo llamado respaldo.sql, ubicado en la ruta especificada.

La razón principal para programar respaldos automáticos cada 15 días en lugar de con mayor frecuencia es encontrar un equilibrio entre la frecuencia de los respaldos y el impacto en el rendimiento del sistema y el almacenamiento.

Consideraciones detrás de esta decisión:

Reducción del impacto en el rendimiento: Los respaldos pueden ser una operación intensiva en recursos, especialmente en bases de datos grandes. Realizarlos con demasiada frecuencia puede afectar el rendimiento del sistema y ralentizar otras operaciones de base de datos que se están ejecutando concurrentemente.

Optimización del espacio de almacenamiento: Los respaldos ocupan espacio en el disco. Realizarlos con demasiada frecuencia puede acumular una gran cantidad de datos de respaldo y consumir espacio en el almacenamiento, especialmente si no se gestionan adecuadamente. Programarlos cada 15 días ayuda a controlar este crecimiento y a gestionar eficazmente el almacenamiento.

Equilibrio entre la frecuencia y la protección de datos: Si bien es importante realizar respaldos con la suficiente frecuencia para proteger los datos críticos de pérdidas, hacerlo cada 15 días sigue siendo un intervalo razonable para la mayoría de los casos de uso. Proporciona una buena protección contra pérdidas de datos significativas sin abrumar al sistema con respaldos frecuentes.

Base de Datos NoSQL

- **Documentación del Esquema: Paciente Donatario**


```
{
 "_id": {
  "$oid": "661ee9784b5411866b113991"
 },
 "paciente_donatario": [
  {
   "_id": "ObjectId",
   "paciente_ID": "int",
   "nombre": "string",
   "apellido": "string",
   "edad": "int",
   "genero": "string",
   "grupo_sanguineo": "string",
   "tipo_sanguineo": "string",
   "fecha_nacimiento": "Date",
   "solicitud_ID": "int",
   "fecha_trasplante": "Date",
   "tipo_trasplante": "string",
   "personal_medico_ID": "int",
   "cirujano_principal": "string",
   "formulario": "int",
   "datos_post_trasplante": {
    "fecha_seguimiento": "Date",
```

```
      "sintomas": [
        {
          "nombre": "string",
          "descripcion": "string",
          "gravedad": "string"
        }
      ]
    }
  }
 ]
}
```


Descripción

Este esquema describe la estructura de los pacientes que actúan como donatarios en el proceso de trasplante de órganos. Contiene información detallada sobre los pacientes, incluidos los datos personales, la información relacionada con la solicitud de trasplante y el seguimiento post-trasplante.

Campos

1. _id: Identificador único del paciente donatario.
2. paciente_ID: Identificador del paciente en la base de datos (entero).
3. nombre: Nombre del paciente (cadena de texto).
4. apellido: Apellido del paciente (cadena de texto).
5. edad: Edad del paciente (entero).
6. genero: Género del paciente (cadena de texto).
7. grupo_sanguineo: Grupo sanguíneo del paciente (cadena de texto).
8. tipo_sanguineo: Tipo sanguíneo del paciente (cadena de texto).
9. fecha_nacimiento: Fecha de nacimiento del paciente (fecha).
10. solicitud_ID: Identificador de la solicitud de trasplante asociada al paciente (entero).
11. fecha_trasplante: Fecha del trasplante (fecha).
12. tipo_trasplante: Tipo de trasplante realizado (cadena de texto).
13. personal_medico_ID: Identificador del personal médico involucrado en el trasplante (entero).
14. cirujano_principal: Nombre del cirujano principal a cargo del trasplante (cadena de texto).
15. formulario: Identificador del formulario asociado al paciente (entero).
16. datos_post_trasplante: Objeto que contiene información sobre el seguimiento post-trasplante.

- fecha_seguimiento: Fecha del seguimiento post-trasplante (fecha).
- sintomas: Lista de síntomas presentados por el paciente después del trasplante.
    - nombre: Nombre del síntoma (cadena de texto).
    - descripcion: Descripción del síntoma (cadena de texto).
    - gravedad: Gravedad del síntoma (cadena de texto).

Uso

Este esquema puede ser utilizado para almacenar y consultar información sobre los pacientes que actúan como donatarios en el proceso de trasplante de órganos. Contiene detalles importantes sobre los pacientes y su historial médico relacionado con el trasplante.

- **Documentación del Esquema: Paciente Donador**


```
{
 "_id": {
  "$oid": "661eeb224b5411866b113994"
 },
 "paciente_donador": {
  "_id": "ObjectId",
  "paciente_ID": "int",
  "nombre": "string",
  "apellido": "string",
  "edad": "int",
  "genero": "string",
  "tipo_sangre": "string",
  "organos_ID": "int",
  "organos_donados": [
   {
    "nombre_organo": "string",
    "fecha_donacion": "datetime",
    "paciente_ID": "int",
    "receptor": {
     "nombre": "string",
     "apellido": "string",
```

```
        "edad": "int",
        "genero": "string",
        "tipo_sangre": "string"
      }
    }
  ]
 }
}
```


Descripción

Este esquema describe la estructura de los pacientes que actúan como donadores en el proceso de trasplante de órganos. Contiene información detallada sobre los donadores, incluidos los datos personales, la información relacionada con la donación de órganos y los receptores de los órganos donados.

Campos

1. _id: Identificador único del paciente donador.
2. paciente_ID: Identificador del paciente en la base de datos (entero).
3. nombre: Nombre del paciente donador (cadena de texto).
4. apellido: Apellido del paciente donador (cadena de texto).
5. edad: Edad del paciente donador (entero).
6. genero: Género del paciente donador (cadena de texto).
7. tipo_sangre: Tipo de sangre del paciente donador (cadena de texto).
8. organos_ID: Identificador del órgano donado (entero).
9. organos_donados: Lista de órganos donados por el paciente.
    - nombre_organo: Nombre del órgano donado (cadena de texto).
    - fecha_donacion: Fecha de la donación del órgano (fecha y hora).
    - paciente_ID: Identificador del paciente receptor del órgano donado (entero).
    - receptor: Detalles del paciente receptor del órgano donado.
        - nombre: Nombre del receptor del órgano (cadena de texto).
        - apellido: Apellido del receptor del órgano (cadena de texto).
        - edad: Edad del receptor del órgano (entero).
        - genero: Género del receptor del órgano (cadena de texto).
        - tipo_sangre: Tipo de sangre del receptor del órgano (cadena de texto).

Uso

Este esquema puede ser utilizado para almacenar y consultar información sobre los pacientes que actúan como donadores en el proceso de trasplante de órganos. Contiene detalles importantes sobre los donadores, los órganos donados y los receptores de los órganos donados.

1. Crear un usuario como administrador

```
> use admin
  db.createUser({
    user: "adminUser",
    pwd: "adminPassword",
    roles: ["root"]
  })
```

2. Crear un usuario para la base de datos específica:

```
> use bd_hospital_kics
  db.createUser({
    user: "regularUser",
    pwd: "regularPassword",
    roles: [{ role: "readWrite", db: "bd_hospital_kics" }]
  })
```

3. Crear un rol personalizado para los documentos de paciente_donatario:

```
> use bd_hospital_kics
  db.createRole({
    role: "donatarioRole",
    privileges: [
      { resource: { db: "myDatabase", collection: "paciente_donatario" }, actions: ["find", "insert", "update", "remove"] }
    ],
    roles: []
  })
```

4. Crear un rol personalizado para los documentos de paciente_donador:

```
> use bd_hospital_kics
  db.createRole({
    role: "donadorRole",
    privileges: [
      { resource: { db: "myDatabase", collection: "paciente_donador" }, actions: ["find", "insert", "update", "remove"] }
    ],
    roles: []
  })
```

5. Asignar roles a los usuarios:

```
> use bd_hospital_kics
  db.grantRolesToUser("regularUser", ["donatarioRole"])
  db.grantRolesToUser("regularUser", ["donadorRole"])
```