Universidad de La Laguna

ADBD Gestión Hospitalaria

JAVIER GONZÁLEZ DE LA BARREDA ARIMANY

JUAN DIEGO RENDÓN CACHAFEIRO

SAMUEL TOLEDO HERNÁNDEZ

Índice:

Enunciado:

1. Título del proyecto práctico final de la base de datos que vas a desarrollar.
2. Objetivos del proyecto. Indique claramente cuál es el propósito que se desea satisfacer.
3. Describir el contexto de la base de datos. Esto es, la especificación de los requisitos. Es obligatorio que sea una propuesta original, puede ser de algún caso real o simplemente supuesto. Los requisitos deben contemplar todos los modelos de datos utilizados en la asignatura: entidades débiles, relaciones triples, tipos IS_A, relaciones 1:N, M:M, y al menos uno de los casos: inclusión, inclusividad, exclusión, exclusividad.
4. Diseño conceptual de la base de datos anteriormente descrita utilizando el modelo entidad-relación e indicar los supuestos semánticos complementarios que consideres oportunos para justificar todas las decisiones del diseño.
5. Obtener el grafo relacional, señalando las claves primarias, las ajenas y las claves alternativas con sus opciones, así como los dominios y las restricciones que consideres oportunas.
6. Describir la base de datos en SQL e implementarla en el Sistema Gestor de Bases de Datos Relacional PostgreSQL en la máquina virtual de la asignatura.
7. Realizar una carga de datos de ejemplo en la base de datos.
8. Diseñar consultas de ejemplo que sirvan para testear la base de datos creada.
9. Implementar un API REST mediante Flask (u otra herramienta de su elección) que permita realizar operaciones CRUD sobre la base de datos PostgreSQL creada.

Objetivos

Gestión Eficiente de Pacientes:

Registrar de manera precisa y accesible la información de los pacientes, incluyendo historiales médicos, citas y datos personales, para facilitar un manejo eficiente de la atención médica.

Proporcionar un sistema que permita el seguimiento del estado de los pacientes, desde su ingreso hasta su egreso, registrando diagnósticos, tratamientos y evolución médica.

Optimización de Recursos Médicos, Inventario de Medicamentos y Equipos Médicos:

Gestionar la asignación de médicos, enfermeros y otros recursos médicos de manera efectiva, permitiendo una distribución equitativa de la carga de trabajo y optimizando los tiempos de consulta.

Mantener un control detallado del inventario de medicamentos y equipos médicos, asegurando la disponibilidad de suministros esenciales y facilitando la gestión de pedidos y reabastecimiento.

Control de Citas y Agenda Médica, Gestión de Facturación:

Facilitar la programación y seguimiento de citas médicas, asegurando una gestión eficiente de la agenda de médicos y la atención oportuna de los pacientes.

Implementación y Restricciones:

El Hospital XYZ, una institución médica de tamaño medio que atiende a una variada población de pacientes, requiere un sistema que gestione de manera eficiente la información detallada de cada individuo. Los registros incluirán datos personales y antecedentes médicos esenciales, como nombre completo, fecha de nacimiento, así como detalles relevantes sobre la historia clínica. La gestión eficiente de la información detallada de los pacientes es fundamental para el correcto funcionamiento de un sistema de atención médica. En este contexto, se requiere diseñar un sistema que cumpla con los siguientes requisitos:

- Se deberá mantener un registro exhaustivo de datos personales y antecedentes médicos de cada paciente atendido. Estos datos incluirán; nombre completo, fecha de nacimiento, historial médico, teléfono y DNI.
- Para poder acceder a los servicios del hospital, los pacientes deberán contar con su cita correspondiente. Para poder garantizar un horario estas deberán tener fecha, un código único y un motivo de consulta.
- Los pacientes pueden ser atendidos tanto por enfermeros como por médicos o ambos. Estos a su vez tienen almacenado su nombre, su especialidad o el número colegiado como identificador.
- Los médicos tienen la capacidad de recetar medicamentos a los pacientes a través de una receta médica, que tiene tanto un código identificador único, una fecha y la cantidad del medicamento recetado. A su vez el medicamento cuenta con su tipo de medicamento, su código identificador único, su precio y el nombre de este.
- Para poder atender a su paciente, los médicos cuentan con diversos equipos médicos de los cuales se sabe su Precio, Tipo de equipo, un atributo de identificación y el nombre del equipo.
- El Hospital XYZ cuenta a su vez con unidades médicas, creadas para poder subdividir al equipo médico y los propios médicos en diferentes sectores (Psiquiatría, Respiratoria o Cardíaca) para mejorar la organización. Para ello además las unidades portan con un código identificatorio y una localización.
- El sistema permitirá la programación eficiente de citas médicas para los pacientes, facilitando una atención ordenada y oportuna aunque debido a que el servicio del Hospital no está disponible durante todo el día los pacientes solo pueden pedir cita en un horario determinado.
- Para garantizar una atención médica equitativa, el sistema deberá gestionar de manera eficiente la asignación de médicos y enfermeras. Esto incluirá la administración de horarios y cargas de trabajo para optimizar la disponibilidad de personal médico.
- Dada la naturaleza sensible de la información médica, se implementará un sistema de acceso seguro y restringido. Esto garantizará la confidencialidad de los datos del paciente, asegurando el cumplimiento de normativas de privacidad y protección de la información. Se implementará una clave no primaria tanto en Paciente como en

Enfermero que contendrá el DNI cifrado, a modo de simular un cifrado de clave para su seguridad.

- Los médicos son capaces de recetar uno o varios Medicamento, del que se conoce el tipo, el código, el precio y el nombre, a través de una Receta Médica de la cual se conoce a su vez la fecha de la receta y su código. Además en cada Receta se especifica la cantidad de cada Medicamento.
- Cada receta es única gracias a que tiene un código de receta que la hace diferente de las demás, pero varias recetas con códigos diferentes pueden estar formadas por el mismo número y tipo de medicamentos. Interpretamos que esta restricción es semántica, por lo cual queda reflejada su implementación en las propias tablas de la base de datos.

Modelo Entidad Relación:

En cuanto a nuestro Modelo Entidad-Relación (https://drive.google.com/file/d/1bPkWoi6SEGCGjA9DZR1NL7mFlP93QURC/view?usp=sharing) se aprecia todas las entidades creadas como puede ser CITA o ENFERMERO. En el Modelo también se observa dos entidades débiles: UNIDAD MEDICA y CITA. Además existe una relación ternaria entre PACIENTE, MEDICO y ENFERMERO, que representa que un paciente pude ser atendido tanto por un enfermero como por un médico. El caso de generalización representa los tipos que pueden tener las unidades médicas.

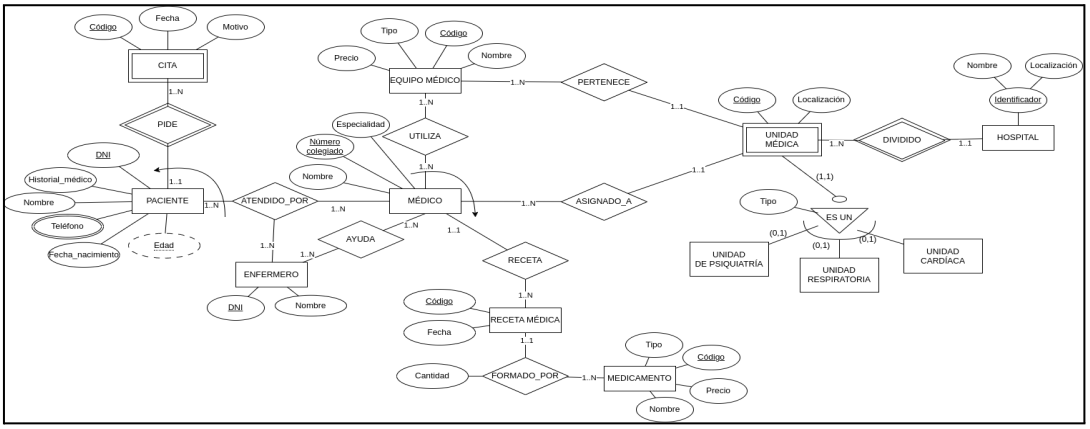

Diagrama: Diagrama de Tablas

Modelo Relacional:

Una vez comentado el Modelo Entidad Relación, podremos entender el Modelo Relacional. Dentro del repositorio de GitHUB se encuentra una carpeta /grafo relacional donde se encuentran ambas imágenes del grafo. Tras seguir los diferentes pasos de transformación del modelo entidad-relación al relacional, se han creado nuevas tablas, TELEFONO, AYUDA, UTILIZA y ATENDIDO_POR.

CITA(Código, *DNI_paciente*, Fecha, Motivo)

PACIENTE(DNI, Historial_medico, Nombre, Fecha_nacimiento, Edad)

MEDICO(Numero_colegiado, Especialidad, Nombre, *Código_unidad_médica, Nombre_hospital, Localizacion_hospital*)

EQUIPO_MEDICO(Código, Nombre, Tipo, Precio, *Código_unidad_médica*)

ENFERMERO(DNI, Nombre)

AYUDA(*DNI_enfermero*, *Numero_colegiado*)

UTILIZA(*Numero_colegiado*, *Código_equipo_medico*)

RECETA_MEDICA(Codigo, *Número_colegiado*, Fecha)

MEDICAMENTO(Codigo, *Código_receta* Tipo, Precio, Nombre, Cantidad)

UNIDAD_MEDICA(Codigo, *Nombre_hospital*, *Localizacion_hospital*, Localizacion, Tipo)

HOSPITAL(Nombre, Localizacion)

TELEFONO(*DNI_paciente*, Teléfono)

ATENDIDO_POR(*DNI_Paciente*, *Numero_colegiado*, *DNI_enfermero*)

UNIDAD_DE_PSIQUIATRÍA(*Codigo_unidad*, *Nombre_hospital*, *Localizacion_hospital*)

UNIDAD_RESPIRATORIA(*Codigo_unidad*, *Nombre_hospital*, *Localizacion_hospital*)

UNIDAD_CARDÍACA(*Codigo_unidad*, *Nombre_hospital*, *Localizacion_hospital*)

Tablas y Diagrama principal:

Mediante una de las aplicaciones o herramientas que hemos utilizado para poder realizar el proyecto “DataGrip” se ha creado un Diagrama de la Base de datos **HOSPITAL XYZ**.

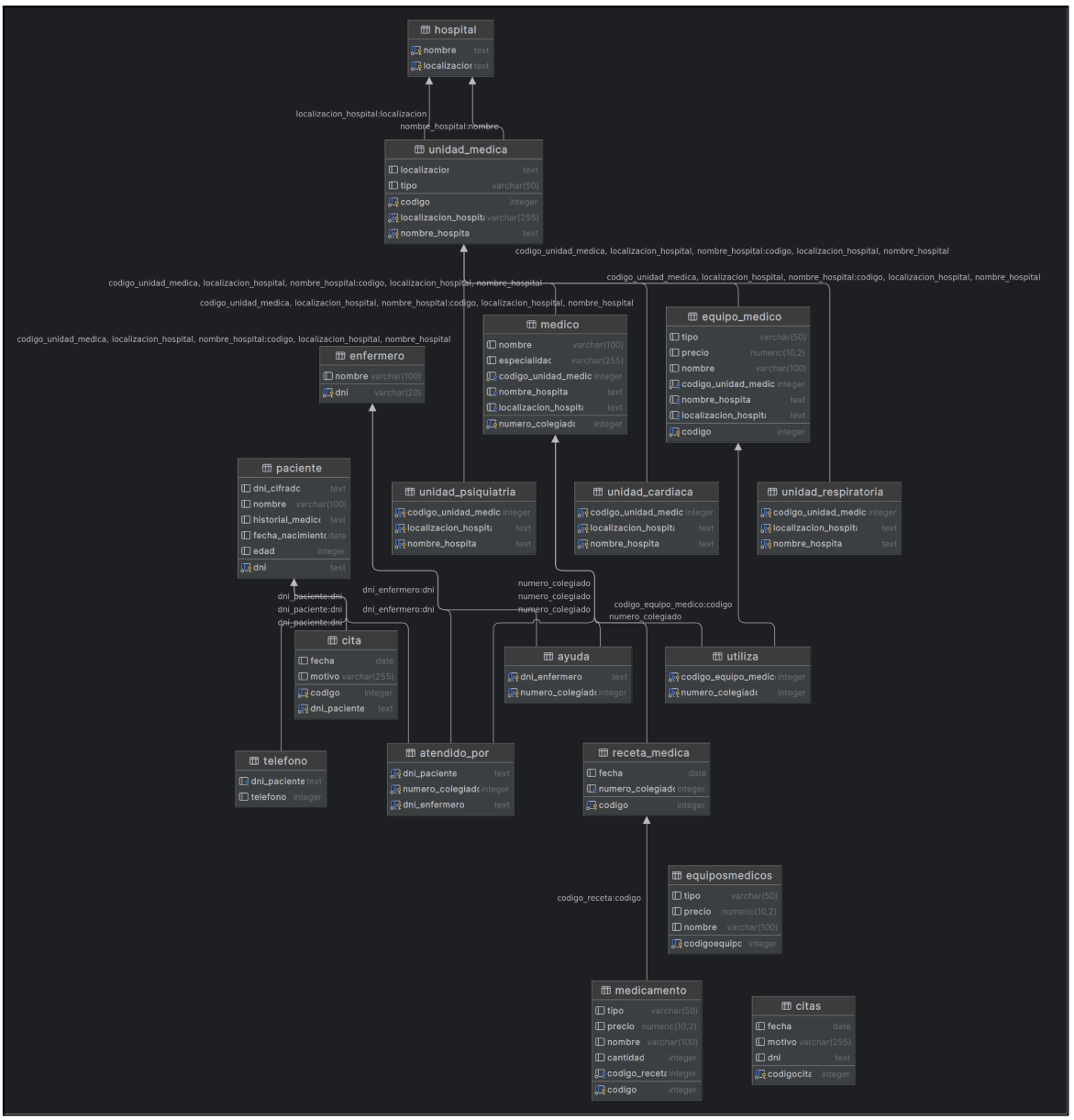

Diagrama: Diagrama de Tablas

Triggers y procedimientos Almacenados:

Cabe destacar que tal y como se comenta en los requisitos, a través de los triggers implementados se hace posible el hecho de almacenar automáticamente la clave cifrada tanto de Pacientes como de Enfermero en DNI cifrado y viceversa, las claves pasarán de cifradas a descifradas a través de un procedimiento almacenado pensado para utilizarse en una consulta como se muestra en la imagen.

```
-- -- Crear la función de encriptación
CREATE OR REPLACE FUNCTION CifrarDNITriggerFunction()
RETURNS TRIGGER AS $$
    BEGIN
--    -- Utilizar la función de encriptación pgp_sym_encrypt con una clave secreta
NEW.dni_cifrado := pgp_sym_encrypt(NEW.DNI, 'clave_secreta');
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Imagen :Función cifrado Pacientes

```
-- Crear la función de encriptación para Enfermero
CREATE OR REPLACE FUNCTION CifrarDNIEnfermeroTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    -- Utilizar la función de encriptación pgp_sym_encrypt con una clave secr
    NEW.dni_cifrado := pgp_sym_encrypt(NEW.DNI, 'clave_secreta');
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Imagen :Función cifrado Enfermero

Además existe un trigger para poder descartar todas las citas que no han sido creadas con una fecha válida, es decir una fecha menor al día actual.

```
-- Crear la función y el trigger para comprobar la fecha de la cita
CREATE OR REPLACE FUNCTION VerificarFechaCita()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.Fecha <= CURRENT_DATE THEN
        RAISE EXCEPTION 'La fecha de la cita debe ser posterior a hoy';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Imagen :Verificar fecha

UPDATES & DELETES

En el apartado de UPDATES y DELETES para comprobar el funcionamiento de las tablas y de la propia base de datos, hemos realizado las siguientes operaciones de prueba:

Utilizando en específico la Tabla de Citas, aunque se puede utilizar de igual manera tanto la tabla Pacientes o Enfermero que son aquellas tablas que cuentas con asserts, podemos comprobar su contenido mediante SELECT * FROM CITAS; (como se observa en la imagen). Como ya se ha comentado anteriormente la tabla CITAS posee un assert que comprueba si las fechas(atributo) son válidas.

	codigo	fecha	motivo	dni_paciente
1	1	2025-03-10	Consulta de rutina	11111111C
2	2	2025-04-05	Consulta de rutina	22222222D
3	3	2025-05-20	Consulta de rutina	33333333E
4	4	2025-06-15	Consulta de rutina	44444444F
5	5	2025-07-25	Consulta de rutina	55555555G
6	6	2025-08-30	Consulta de rutina	66666666H
7	7	2025-10-18	Consulta de rutina	77777777I

Imagen :Tabla de Citas

Si sobre esta tabla decidimos realizar una serie de Updates tal y como se muestra en la imagen , comprobaremos que el primero de estos se ejecuta correctamente, en cambio el segundo posee una fecha inválida “fecha = 2020-03-10”, por lo cual saltará un “RAISE EXCEPTION 'La fecha de la cita debe ser posterior a hoy';“

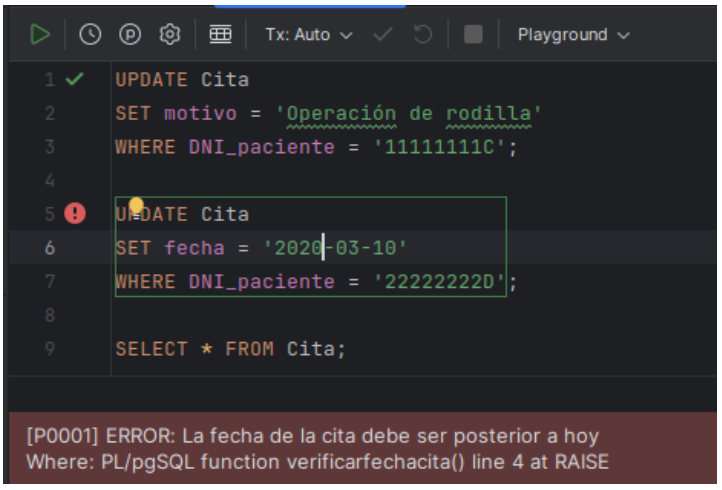

Imagen :Updates en Citas

Para poder realizar dicha operación debemos introducir una fecha válida para realizar el UPDATE, como puede ser “fecha = 2027-03-10”para poder actualizarla correctamente.

	codigo	fecha	motivo	dni_paciente
1	3	2025-05-20	Consulta de rutina	33333333E
2	4	2025-06-15	Consulta de rutina	44444444F
3	5	2025-07-25	Consulta de rutina	555555556
4	6	2025-08-30	Consulta de rutina	66666666H
5	7	2025-10-18	Consulta de rutina	77777777I
6	1	2025-03-10	Operación de rodilla	11111111C
7	2	2027-03-10	Consulta de rutina	22222222D

Imagen :Tabla de Citas

Para terminar en la imagen , podemos apreciar una consulta de DELETE y correspondiente tabla de Citas

```
DELETE FROM Cita
WHERE Codigo = 3 AND DNI_paciente = '33333333E';

SELECT * FROM Cita;
```

	codigo	fecha	motivo	dni_paciente
1	4	2025-06-15	Consulta de rutina	44444444F
2	5	2025-07-25	Consulta de rutina	555555556
3	6	2025-08-30	Consulta de rutina	66666666H
4	7	2025-10-18	Consulta de rutina	77777777I
5	1	2025-03-10	Operación de rodilla	11111111C
6	2	2027-03-10	Consulta de rutina	22222222D

Imagen :Delete y Tabla de Citas