

# HOSPITAL SAN ROQUE

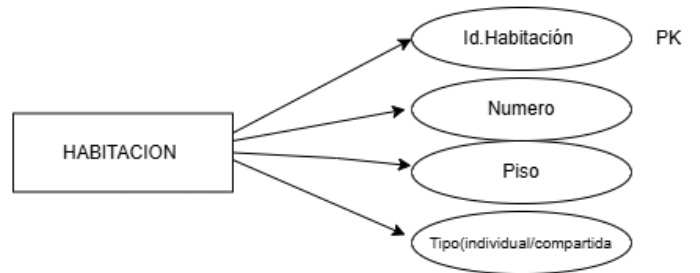
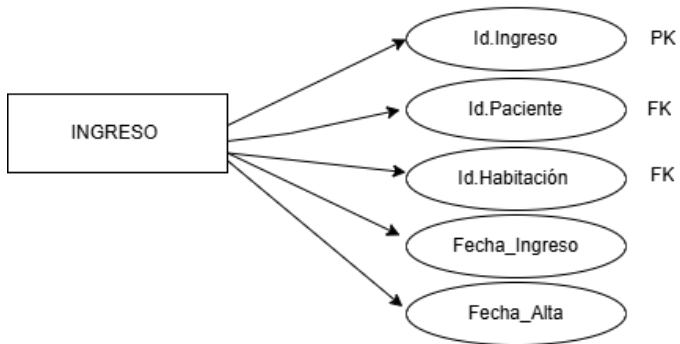
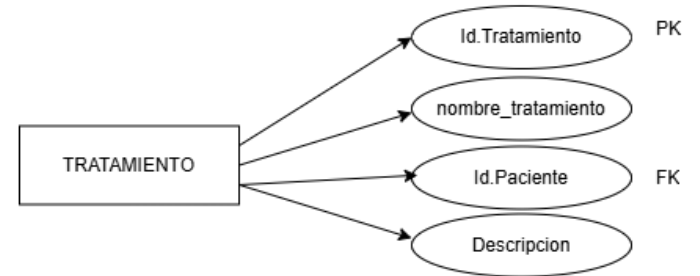
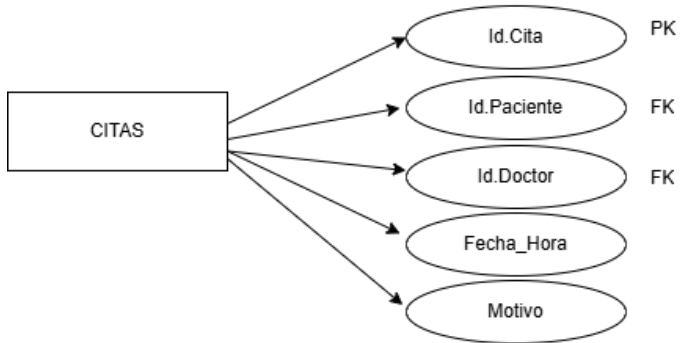
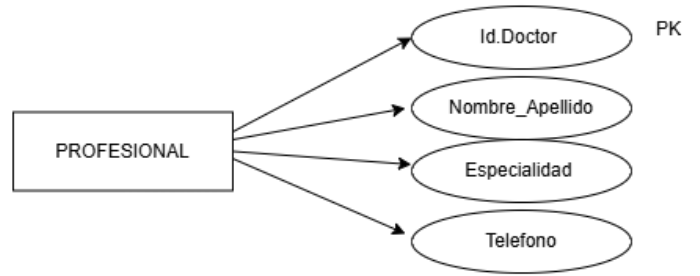
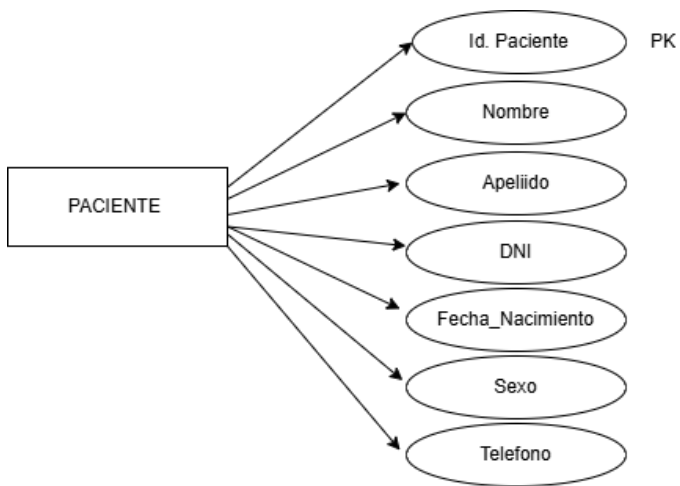
El Hospital San Roque SA. es un establecimiento medico de consultorios e internación; adquirido hace poco tiempo por una nueva firma y requiere una nueva base de datos para su mayor comodidad y orden para sus profesionales y pacientes.

La base de datos que ellos requieren, es una orden detallada día a día de todas sus consultas, internaciones y tratamientos médicos realizados.

## Índice

- [Diagrama ER](#)
- [Lista de Tablas](#)
  - [Triggers](#)
- [Store Procedure](#)
  - [Vistas](#)
- [Funciones](#)

## Diagrama Entidad-Relación



## Lista de tablas

**Tabla Paciente:** En esta tabla encontraremos los datos del paciente.

Id. Paciente (PRIMARY KEY) - Nombre – Apellido – Dni - Fecha\_nacimiento – Sexo – Teléfono

**Tabla Profesional:** En esta tabla encontraremos los datos del doctor/profesional a cargo y su especialidad.

Id. Doctor (PRIMARY KEY) – Nombre\_Apellido – Especialidad – Telefono

**Tabla Citas:** En esta tabla encontraremos los datos de la consulta del paciente; como fecha, hora y motivo de la cita.

Id. Cita (PRIMARY KEY) – Id. Paciente (FOREIGN KEY) – Id. Doctor (FOREIGN KEY) – Fecha\_hora – Motivo

**Tabla Tratamiento:** En esta tabla encontraremos la/s patologías y diagnóstico del paciente.

Id. Tratamiento (PRIMARY KEY) – Nombre tratamiento - Id. Paciente (FOREIGN KEY) – Descripción

**Tabla Ingreso:** En esta tabla encontraremos todo lo relacionado al ingreso del paciente en caso de internación, fecha y alta del mismo.

Id. Ingreso (PRIMARY KEY) – Id. Paciente (FOREIGN KEY) – Id. Habitación (FOREIGN KEY) – Fecha\_ingreso – Fecha\_alta

**Tabla Habitación:** En esta tabla encontraremos los datos de la habitación del paciente internado.

Id. Habitación (PRIMARY KEY) – Id Ingreso (FOREIGN KEY) – Piso – Número\_hab  
– Tipo (individual/compartida)

<b><i>PACIENTE</i></b>		
<b>CAMPO</b>	<b>TIPO DE DATO</b>	<b>TIPO DE CLAVE</b>
ID_PACIENTE	INT AUTO_INCREMENT NOT NULL	PRIMARY KEY
NOMBRE	VARCHAR NOT NULL	
APELLIDO	VARCHAR NOT NULL	
DNI	INT UNIQUE NOT NULL	
FECHA_NACIMIENTO	DATE NOT NULL	
SEXO	VARCHAR NOT NULL	
TELEFONO	VARCHAR NOT NULL	

<b><i>PROFESIONAL</i></b>		
<b>CAMPO</b>	<b>TIPO DE DATO</b>	<b>TIPO DE CLAVE</b>
ID_DOCTOR	INT AUTO_INCREMENT NOT NULL	PRIMARY KEY
NOMBRE_APELLIDO	VARCHAR NOT NULL	
ESPECIALIDAD	VARCHAR NOT NULL	
TELEFONO	VARCHAR NOT NULL	

<b><i>CITAS</i></b>		
<b>CAMPO</b>	<b>TIPO DE DATO</b>	<b>TIPO DE CLAVE</b>
ID_CITA	INT AUTO_INCREMENT NOT NULL	PRIMARY KEY
ID_PACIENTE	INT NOT NULL	FOREIGN KEY
ID_DOCTOR	INT NOT NULL	FOREIGN KEY
FECHA_HORA	DATETIME DEFAULT CURRENT_TIMESTAMP	
MOTIVO	VARCHAR NOT NULL	

<b>TRATAMIENTO</b>		
<b>CAMPO</b>	<b>TIPO DE DATO</b>	<b>TIPO DE CLAVE</b>
ID_TRATAMIENTO	INT NOT NULL	PRIMARY KEY
NOMBRE_TRATAMIENTO	VARCHAR NOT NULL	
ID_PACIENTE	INT NOT NULL	FOREIGN KEY
DESCRIPCION	VARCHAR NOT NULL	

<b>INGRESO</b>		
<b>CAMPO</b>	<b>TIPO DE DATO</b>	<b>TIPO DE CLAVE</b>
ID_INGRESO	INT NOT NULL	PRIMARY KEY
ID_PACIENTE	INT NOT NULL	FOREIGN KEY
ID_HABITACION	INT NOT NULL	
FECHA_INGRESO	DATETIME NOT NULL	
FECHA_ALTA	DATETIME NOT NULL	

<b>HABITACION</b>		
<b>CAMPO</b>	<b>TIPO DE DATO</b>	<b>TIPO DE CLAVE</b>
ID_HABITACION	INT NOT NULL	PRIMARY KEY
ID_INGRESO	INT NOT NULL	FOREIGN KEY
PISO	VARCHAR NOT NULL	
NUMERO_HAB	INT NOT NULL	
TIPO (individual/compatida)	VARCHAR NOT NULL	

## TRIGGERS

Trigger creado para evitar que un paciente tenga dos ingresos a la vez.

```
DELIMITER //
```

```
CREATE TRIGGER evitar_ingreso_duplicado
BEFORE INSERT ON ingreso
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1 FROM ingreso
        WHERE id_paciente = NEW.id_paciente AND fecha_alta IS NULL
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El paciente ya está ingresado.';
    END IF;
END //
```

```
DELIMITER ;
```

Trigger creado para evitar agendar dos veces la misma cita.

DELIMITER //

```
CREATE TRIGGER evitar_cita_duplicada
BEFORE INSERT ON citas
FOR EACH ROW
BEGIN
    IF EXISTS (
        SELECT 1 FROM citas
        WHERE id_paciente = NEW.id_paciente
            AND id_doctor = NEW.id_doctor
            AND DATE(fecha_hora) = DATE(NEW.fecha_hora)
    ) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El paciente ya tiene una cita con este médico ese día.';
    END IF;
END //
```

DELIMITER ;

Trigger creado para controlar que no se repitan tratamientos innecesariamente.

DELIMITER //

```
CREATE TRIGGER evitar_tratamiento_duplicado
BEFORE INSERT ON tratamiento
FOR EACH ROW
```

BEGIN

IF EXISTS (

SELECT 1 FROM tratamiento

WHERE id\_paciente = NEW.id\_paciente

AND nombre\_tratamiento = NEW.nombre\_tratamiento

) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'El paciente ya tiene este tratamiento asignado.';

END IF;

END //

DELIMITER ;

### STORE PROCEDURE

Store procedure para agendar una nueva cita.

DELIMITER //

CREATE PROCEDURE Registrar\_Cita (

IN p\_id\_paciente INT,

IN p\_id\_doctor INT,

IN p\_fecha DATETIME,

IN p\_motivo VARCHAR(30)

)

BEGIN

INSERT INTO CITAS (id\_paciente, id\_doctor, fecha\_hora, motivo)



```
VALUES (p_id_paciente, p_id_doctor, p_fecha, p_motivo);  
END //
```

```
DELIMITER ;
```

Store procedure creado para dar de alta a un paciente.

```
DELIMITER //
```

```
CREATE PROCEDURE Dar_Alta_Paciente (  
    IN p_id_ingreso INT,  
    IN p_fecha_alta DATETIME  
)
```

```
BEGIN
```

```
    UPDATE ingreso
```

```
    SET fecha_alta = p_fecha_alta
```

```
    WHERE id_ingreso = p_id_ingreso;
```

```
END //
```

```
DELIMITER ;
```

Store procedure creado para ingresar paciente a habitación.

```
DELIMITER //
```

```
CREATE PROCEDURE Ingresar_Paciente (  
    IN p_id_paciente INT,  
    IN p_id_habitacion INT,
```

```

    IN p_fecha_ingreso DATETIME
)
BEGIN
    INSERT INTO ingreso (id_paciente, id_habitacion, fecha_ingreso, fecha_alta)
    VALUES (p_id_paciente, p_id_habitacion, p_fecha_ingreso, NULL);
END //

DELIMITER ;

```

Store procedure creado para ingresar pacientes nuevos al hospital

```

DELIMITER //

CREATE PROCEDURE ingresar_nuevo_paciente (
    IN p_nombre VARCHAR(20),
    IN p_apellido VARCHAR(20),
    IN p_dni INT,
    IN p_fecha_nacimiento DATE,
    IN p_sexo VARCHAR(2),
    IN p_telefono VARCHAR(15)
)
BEGIN
    -- Verificar que el DNI no esté repetido
    IF EXISTS (SELECT 1 FROM paciente WHERE dni = p_dni) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Ya existe un paciente con ese DNI.';
    END IF;
END //

```

ELSE

INSERT INTO paciente (

nombre, apellido, dni, fecha\_nacimiento, sexo, telefono

) VALUES (

p\_nombre, p\_apellido, p\_dni, p\_fecha\_nacimiento, p\_sexo, p\_telefono

);

END IF;

END //

DELIMITER ;

CALL ingresar\_nuevo\_paciente('Lucia', 'Fernandez', 40875123, '1994-07-21', 'F', '3514448899');

CALL ingresar\_nuevo\_paciente('Marcos', 'Torres', 37249865, '1987-02-10', 'M', '3516677885');

CALL ingresar\_nuevo\_paciente('Sofia', 'Herrera', 42315678, '2001-11-05', 'F', '3515511997');

CALL ingresar\_nuevo\_paciente('Pedro', 'Salinas', 38977452, '1975-03-16', 'M', '3518844556');

### VISTAS

Vista creada para ver fácilmente todas las citas con nombres y especialidades.

CREATE VIEW vista\_historial\_citas AS

SELECT

c.id\_cita,

```
c.fecha_hora,  
c.motivo,  
p.nombre AS nombre_paciente,  
p.apellido AS apellido_paciente,  
d.nombre_apellido AS nombre_doctor,  
d.especialidad  
FROM citas c  
JOIN paciente p ON c.id_paciente = p.id_paciente  
JOIN profesional d ON c.id_doctor = d.id_doctor;
```

Vista creada para ver los tratamientos que tiene cada paciente.

```
CREATE VIEW vista_tratamientos_paciente AS  
SELECT  
t.id_tratamiento,  
t.nombre_tratamiento,  
t.descripcion,  
p.nombre,  
p.apellido  
FROM tratamiento t  
JOIN paciente p ON t.id_paciente = p.id_paciente;
```

Vista creada para ver si la habitación esta ocupada o disponible.

```
CREATE VIEW vista_ocupacion_habitaciones AS  
SELECT  
h.id_habitacion,
```

```
h.numero_hab,  
h.piso,  
h.tipo,  
CASE  
    WHEN i.fecha_alta IS NULL THEN 'Ocupada'  
    ELSE 'Libre'  
END AS estado  
FROM habitacion h  
LEFT JOIN ingreso i ON h.id_habitacion = i.id_habitacion  
AND i.fecha_alta IS NULL;
```

## FUNCIONES

Función creada para obtener el nombre completo del paciente por id.

DELIMITER //

```
CREATE FUNCTION nombre_completo_paciente(p_id INT)  
RETURNS VARCHAR(50)  
DETERMINISTIC  
BEGIN  
    DECLARE nombre_completo VARCHAR(50);  
    SELECT CONCAT(nombre, ' ', apellido)  
    INTO nombre_completo  
    FROM paciente
```

```
WHERE id_paciente = p_id;  
RETURN nombre_completo;  
END //
```

```
DELIMITER ;
```

Función creada para ver la cantidad de citas de un paciente.

```
DELIMITER //
```

```
CREATE FUNCTION total_citas_paciente(p_id INT)  
RETURNS INT  
DETERMINISTIC  
BEGIN  
    DECLARE total INT;  
    SELECT COUNT(*) INTO total FROM CITAS WHERE id_paciente = p_id;  
    RETURN total;  
END //
```