Proyecto #2 HospiTEC

Bases de Datos

Profesor: Marco Rivera Meneses

Estudiantes:

Gustavo Adolfo Gamboa Mora 2020023596

Isaac Somarribas Montero 20202125516

María Nicole Valverde Jiménez 2022200481

Emmanuel Esquivel Chavarría 2022312336

Fabián Castillo Cerdas 2020202938

I Semestre 2024

Fecha de entrega: 05-06-2024

Índice

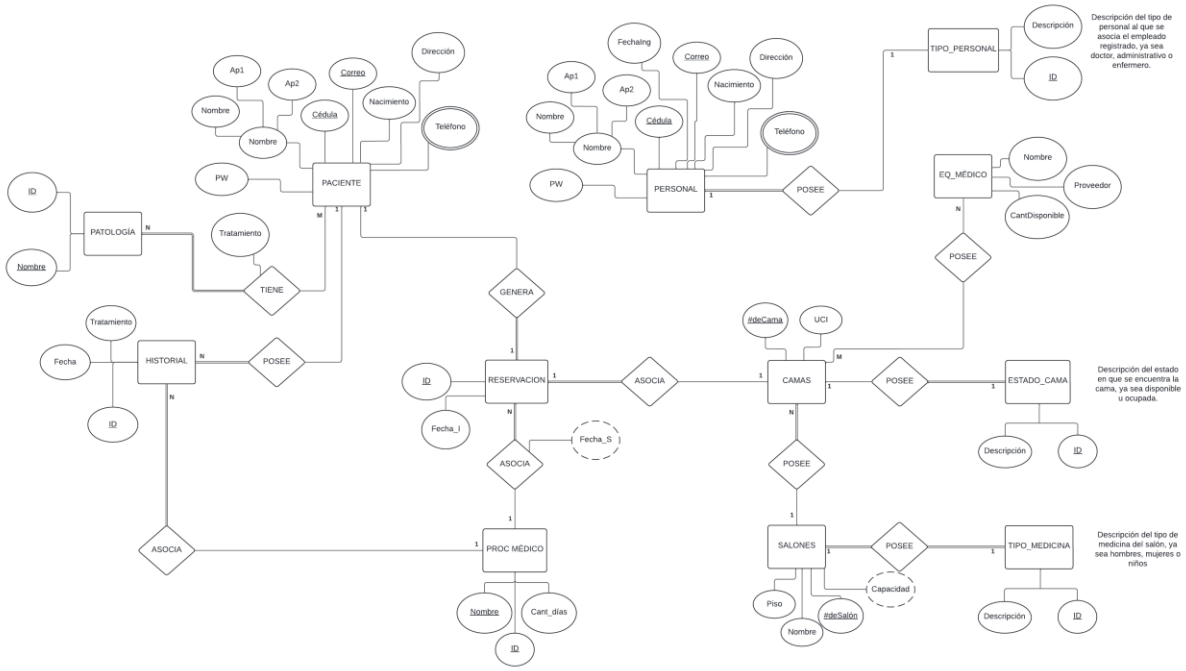
Introducción

Las bases de datos pueden modelar cualquier situación en donde se requiera un manejo de datos bastante preciso y detallado, es por eso que el presente documento explica detalladamente la implementación de una aplicación web y una API con una base de datos relacional en PostgreSQL con el objetivo de modelar la reserva de camas, la gestión de personal, equipo médico, procedimientos médicos, patologías, salones y camas de HospiTEC.

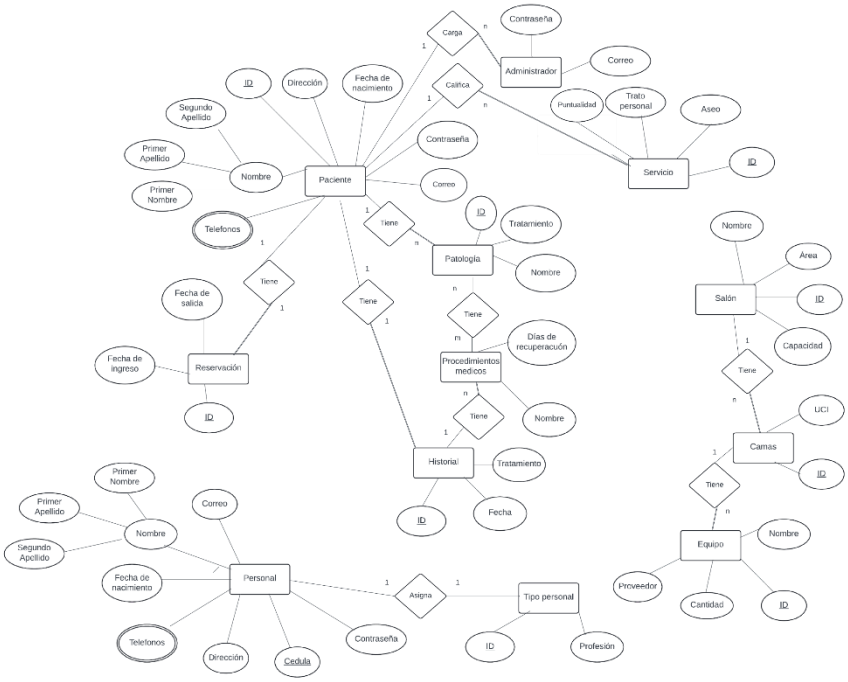
En el documento se detallarán tanto el modelo conceptual como el modelo relacional utilizado para crear la base de datos relacional en PostgreSQL, se detallarán cada una de las tablas que forman parte de la base de datos creada, los procedimientos almacenados, triggers y funciones utilizadas, además del respectivo diagrama de arquitectura en donde se visualizará la manera en que operan cada uno de los componentes de este proyecto y como se conjuntan para la recepción de datos y la visualización de los mismos desde una aplicación web, con la ayuda de una API que centraliza la información y maneja los datos almacenados en la base de datos.

Modelo conceptual utilizando la notación de Chen.

Modelo Conceptual #1

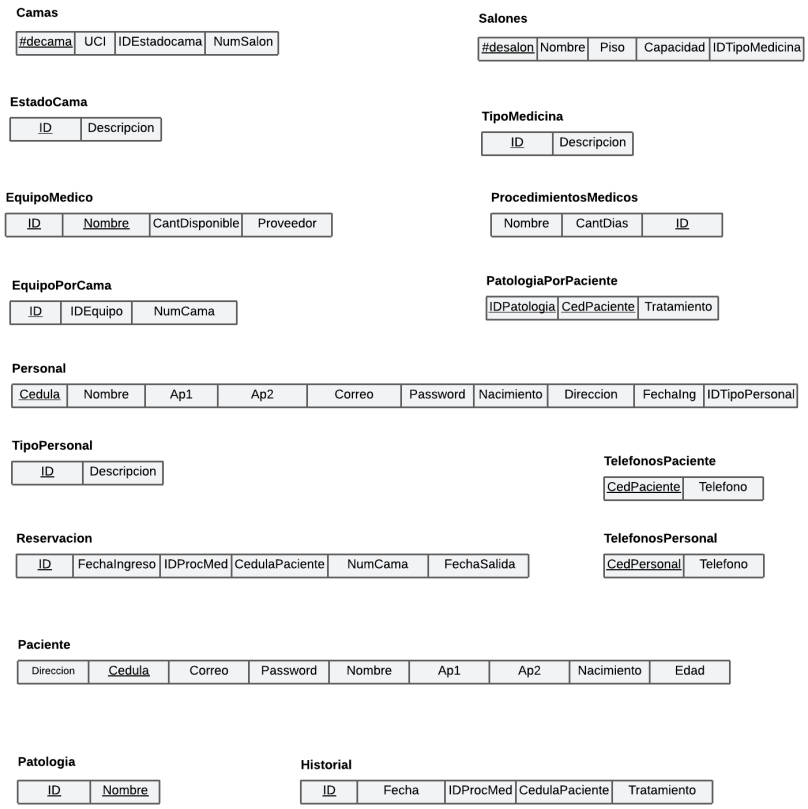


Modelo Conceptual #2



Bajo un análisis exhaustivo se decidió utilizar el modelo conceptual #1, debido a que consideramos que hay un mejor modelo de entidades, a pesar de que ambos están muy bien hechos y cualquiera de los dos pudo haber sido utilizado en el proyecto en sí. El modelo relacional y todo lo demás del proyecto está relacionado con el modelo conceptual #1.

Modelo Relacional



Descripción de las estructuras de datos desarrolladas (Tablas).

Tabla TIPO_PERSONAL

- ID: Identificador único para cada tipo de personal.
- Descripcion: Cadena de texto de hasta 20 caracteres que describe el tipo de personal.

Tabla ESTADO_CAMA

- **ID:** Identificador único de tipo serial para cada estado de cama.
- **Estado:** Describe el estado de la cama.

Tabla TIPO_MEDICINA

- **ID: Identificador único de tipo serial para cada tipo de medicina.**
- **Descripcion: Cadena de texto de hasta 20 caracteres que describe el tipo de medicina.**

Tabla PROCEDIMIENTO_MEDICO

- **ID: Identificador único de tipo serial para cada procedimiento médico.**
- **Nombre: Cadena de texto única de hasta 100 caracteres que describe el nombre del procedimiento.**
- **IDPatologia: Identificador de la patología asociada, tipo entero.**
- **CantDias: Número entero que indica la cantidad de días asociados al procedimiento.**

Tabla PERSONAL

- **Nombre: Cadena de texto de hasta 100 caracteres para el nombre del personal.**
- **AP1: Cadena de texto de hasta 100 caracteres para el primer apellido del personal.**
- **AP2: Cadena de texto de hasta 100 caracteres para el segundo apellido del personal.**
- **Cedula: Número entero único que identifica al personal.**
- **Direccion: Cadena de texto de hasta 200 caracteres para la dirección del personal.**
- **Nacimiento: Fecha de nacimiento del personal.**
- **Correo: Cadena de texto de hasta 100 caracteres para el correo electrónico del personal.**
- **Password: Cadena de texto de hasta 100 caracteres para la contraseña del personal.**
- **FechaIngreso: Fecha de ingreso del personal.**
- **IDTipoPersonal: Identificador del tipo de personal, tipo entero.**
- **PRIMARY KEY: Cedula.**

Tabla TELEFONOS_PERSONAL

- **ID: Identificador único de tipo serial.**
- **PersonalCedula: Número entero que referencia la cédula del personal.**
- **Telefono: Cadena de texto de hasta 20 caracteres que contiene el número de teléfono del personal.**

Tabla PACIENTE

- **Nombre: Cadena de texto de hasta 100 caracteres para el nombre del paciente.**
- **AP1: Cadena de texto de hasta 100 caracteres para el primer apellido del paciente.**
- **AP2: Cadena de texto de hasta 100 caracteres para el segundo apellido del paciente.**
- **Cedula: Número entero único que identifica al paciente.**
- **Nacimiento: Fecha de nacimiento del paciente.**
- **Direccion: Cadena de texto de hasta 200 caracteres para la dirección del paciente.**
- **Correo: Cadena de texto de hasta 100 caracteres para el correo electrónico del paciente.**
- **Password: Cadena de texto de hasta 100 caracteres para la contraseña del paciente.**
- **PRIMARY KEY: Cedula.**

Tabla TELEFONOS_PACIENTE

- **ID: Identificador único de tipo serial.**
- **PacienteCedula: Número entero que referencia la cédula del paciente.**
- **Telefono: Cadena de texto de hasta 20 caracteres que contiene el número de teléfono del paciente.**

Tabla PATOLOGIA

- **ID: Identificador único de tipo serial para cada patología.**
- **Nombre: Cadena de texto única de hasta 50 caracteres que describe el nombre de la patología.**

Tabla PATOLOGIA_POR_PACIENTE

- **ID_Patologia:** Identificador de la patología, tipo entero.
- **CedulaPaciente:** Número entero que referencia la cédula del paciente.
- **Tratamiento: Cadena de texto de hasta 200 caracteres que describe el tratamiento.**

Tabla HISTORIAL_MEDICO

- **ID: Identificador único de tipo serial.**
- **Fecha: Fecha del historial médico.**
- **Tratamiento: Cadena de texto de hasta 200 caracteres que describe el tratamiento.**
- **CedulaPaciente:** Número entero que referencia la cédula del paciente.
- **ID_Procedimiento:** Identificador del procedimiento médico, tipo entero.

Tabla SALON

- **NumSalon:** Número entero único que identifica al salón.
- **Nombre: Cadena de texto de hasta 100 caracteres que describe el nombre del salón.**
- **Piso: Número entero que indica el piso del salón.**
- **ID_Tipo_Medicina:** Identificador del tipo de medicina, tipo entero.

Tabla CAMA

- **NumCama:** Número entero único que identifica a la cama.
- **UCI: Valor booleano que indica si la cama está en la UCI.**
- **ID_EstadoCama:** Identificador del estado de la cama, tipo entero.
- **Num_Salon:** Número entero que referencia el salón al que pertenece la cama.

Tabla EQUIPO_MEDICO

- **ID: Identificador único de tipo serial para cada equipo médico.**
- **Nombre: Cadena de texto única de hasta 100 caracteres que describe el nombre del equipo.**

- **Proveedor: Cadena de texto de hasta 100 caracteres que describe al proveedor del equipo.**
- **CantDisponible:** Número entero que indica la cantidad disponible del equipo.

Tabla EQUIPO_POR_CAMA

- **Num_Cama:** Número entero que referencia la cama.
- **ID_EquipoMedico:** Identificador del equipo médico, tipo entero.

Tabla RESERVACION

- **ID: Identificador único de tipo serial.**
- **FechaIngreso:** Fecha de ingreso del paciente.
- **CedPaciente:** Número entero que referencia la cédula del paciente.
- **IDProcMed:** Identificador del procedimiento médico, tipo entero.
- **NumCama:** Número entero que referencia la cama.

Descripción de los Store Procedures/Funciones/Triggers implementados.

FUNCIONES

Encriptar_password() esta función retorna un trigger en el que el cual encripta la contraseña cuando se añade o se actualiza un paciente o personal.

Encriptar_password(passwords_ Text): esta función encripta la contraseña y retorna el texto encriptado.

validar_reservacion_unica(): retorna un trigger y se encarga de validar que no se realice una reservación de una cama en la misma fecha más de una vez.

eliminar_telefonos_personal(): retorna un trigger y se encarga de que cuando elimine una tupla de personal elimine los teléfonos asociados en la tabla M:N de teléfonos_personal

TRIGGERS

encriptar_password_paciente: se encarga de encriptar los passwords en la tabla paciente cuando se realiza un insert

encriptar_password_personal: se encarga de encriptar los passwords en la tabla personal cuando se realiza un insert

encriptar_password_personal_update: se encarga de encriptar los passwords en la tabla personal cuando se realiza un update

encriptar_password_paciente_update: se encarga de encriptar los passwords en la tabla paciente cuando se realiza un update

trigger_validar_reservacion_unica: se encarga de validar si una reservación es única

tr_eliminar_telefonos_personal: se encarga de cuando se realiza un delete en personal elimine los teléfonos en la table teléfonos_personal

STORE PROCEDURES

crear_personal: este store procedure se encarga de crear una tupla en la tabla personal

update_personal: este store procedure se encarga de actualizar una tupla en la tabla personal

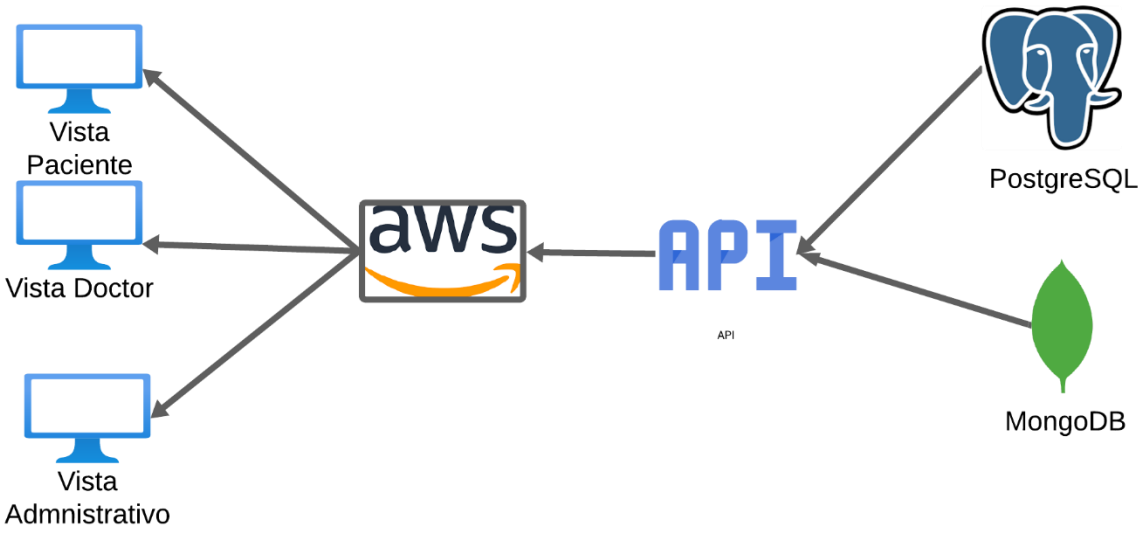
eliminar_personal: este store procedure se encarga de eliminar una tupla en la tabla personal

crear_reservacion: este store procedure se encarga de crear una tupla en la tabla reservación

eliminar_reservacion: este store procedure se encarga de eliminar una tupla en la tabla reservación

transferir_pacientes_desde_temporal: este store procedure se encarga de obtener los datos que se guardaron en la tabla temporal y añadirlos a las tablas paciente, y los teléfonos en la tabla teléfonos_paciente

Descripción detallada de la arquitectura desarrollada.



Descripción de la Arquitectura

Página WEB:

Los usuarios interactúan con la aplicación web a través de un navegador web. La aplicación web está desarrollada con React.

API:

La capa de servicios está desarrollada en C# utilizando .NET Core. Esta capa expone APIs REST que son consumidas por el la WEB en React.

La APIs maneja operaciones CRUD (crear, leer, actualizar, eliminar) para los diferentes componentes del sistema, como pacientes, reservaciones, historiales clínicos, entre otros.

Base de datos:

La base de datos relacional principal es PostgreSQL. Almacena datos estructurados como información de pacientes, reservaciones, procedimientos médicos, historiales clínicos.

La base de datos está normalizada al menos en la tercera forma normal (3NF) para asegurar integridad y reducir redundancia.

MongoDB se utiliza para almacenar evaluaciones del servicio recibido por los pacientes. Esta base de datos NoSQL es adecuada para manejar datos semi-estructurados y no estructurados.

Despliegue en la Nube

La capa de servicios se despliega en la nube utilizando servicios de Azure.

Problemas conocidos

Algunos de los problemas es la carga de pacientes la cual genera errores a la hora de la conexión con la base de datos, por alguna razón mantiene esa conexión abierta.

Problemas encontrados

Descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

Se tuvo problemas a la hora de hacer la carga de pacientes debido a la conexión de la API, la cual no cierra la conexión y la deja abierta.

Otro problema fue la conexión con MongoDB pero fue solucionado satisfactoriamente.

Algunos de los post daban errores inesperados pero esto más que todo fue por un mal manejo de la sintaxis a la hora de escribir los querys, lo cual fue solucionado satisfactoriamente con una repasada a los conceptos de SQL.

Conclusiones del proyecto.

- Se puede concluir que el uso de PostgreSQL es de suma importancia para el almacenamiento de datos en aplicaciones web con una sintaxis similar a SQL server.
- Además, el aplicar correctamente los conceptos y los pasos para realizar los modelos conceptual y relacional de la base de datos ayuda a generar una base de datos de manera más sencilla.
- La herramienta React es muy útil a la hora de realizar aplicaciones web, ya que esta permite que la creación de interfaces sea fácil y predecible.

Recomendaciones del proyecto.

- Se recomienda visualizar antes el manual de usuario de la aplicación web para un mejor manejo de la misma.
- Además, se recomienda visualizar el documento de instalación del proyecto para instalar todo lo necesario para correr de manera satisfactoria el proyecto.
- Utilizar herramientas como React pueden ayudar a crear aplicaciones web.

Bibliografía consultada en todo el proyecto.

C# Coding Conventions (C# Programming Guide). (2018-10-04). Recuperado de: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-aprogram/coding-convention

React Documentation Getting Started – React (reactjs.org)

PostgreSQL: Documentation

MongoDB Documentation