Escuela de Ingeniería en Computadores
CE-3101
Bases de Datos
Grupo 1

Documentación Técnica - HospiTEC

Profesor:
Marco Rivera Meneses

Estudiantes:
Dylan Garbanzo Fallas (2021057775)
Alejandra Rodríguez Castro (2021131070)
Carlos Eduardo Rodríguez Segura (2022437835)
Ricardo Borbón Mena (2021132065)
José María Vindas Ortiz (2022209471)

I Semestre 2024

Introducción

El Hospital Tecnológico surge como una respuesta a la necesidad de mejorar la infraestructura y los servicios médicos ante emergencias nacionales, especialmente después de los desafíos presentados por la pandemia del Covid-19. Esta iniciativa busca transformar la clínica de salud del Instituto Tecnológico en un hospital de vanguardia, capaz de atender cualquier emergencia nacional y ofrecer servicios médicos al público en general, asegurando así su sostenibilidad financiera.

Para cumplir con estos objetivos, es necesario desarrollar un sistema administrador del hospital que integre múltiples funcionalidades, facilitando la gestión eficiente de pacientes, personal, equipo médico, y procedimientos. Este documento detalla el diseño, desarrollo e implementación de una aplicación web que servirá como el núcleo de este sistema administrador, abordando tanto las necesidades de los pacientes como las del personal médico y administrativo.

La aplicación será desarrollada utilizando tecnologías modernas, asegurando robustez, escalabilidad y facilidad de uso. Con bases de datos en PostgreSQL y MongoDB, una capa de servicios desarrollada en C#, y una interfaz web construida con Angular, Bootstrap, HTML5 y CSS, el sistema garantizará un manejo eficiente de la información y una experiencia de usuario óptima.

Objetivos del Proyecto

Objetivo General

Desarrollar una aplicación web integral que facilite la gestión administrativa y operativa del Hospital Tecnológico, permitiendo la administración eficiente de pacientes, personal médico y administrativo, equipo médico, procedimientos y camas hospitalarias, además de proveer herramientas para la evaluación de servicios y generación de reportes.

Objetivos Específicos

➔ Aplicar los conceptos del modelo conceptual y relacional.

➔ Crear una Base de Datos relacional en PostgreSQL para que permita el almacenamiento de los datos.

➔ Crear un servicio API para que centralice la funcionalidad.

➔ Crear una página Web para que exponga la funcionalidad al usuario.

Descripción de los Componentes

Arquitectura Utilizada

La arquitectura del proyecto se estructura en torno a la interacción entre tres componentes principales. En primer lugar, se encuentra la página web, que sirve como interfaz de usuario para acceder a las funcionalidades del sistema. Estas interfaces están diseñadas para ofrecer una experiencia de usuario intuitiva y fluida.

El segundo componente es la API REST o servicio web, que actúa como intermediario entre los clientes (página web y aplicación móvil) y la base de datos. Esta API proporciona procesos que permiten a los clientes realizar diversas operaciones, como crear, leer, actualizar y eliminar datos. Estos servicios están diseñados siguiendo los principios RESTFUL para asegurar una comunicación eficiente y escalable.

Finalmente, la base de datos es el repositorio central de datos del sistema. Aquí se almacena toda la información relevante, como detalles de laboratorios, activos, usuarios y registros de actividad. La base de datos está diseñada para ser robusta y escalable, permitiendo un acceso y manipulación de los datos sencillo mediante consultas SQL.

Esta arquitectura ofrece varios beneficios. En primer lugar, proporciona escalabilidad, ya que cada componente puede escalarse de forma independiente según las necesidades del sistema. Además, la modularidad de la arquitectura facilita el mantenimiento, permitiendo identificar y corregir errores de manera eficiente.

Por último, la arquitectura garantiza la seguridad de los datos, ya que la API REST actúa como un punto de acceso seguro a la base de datos, permitiendo implementar medidas de seguridad como autenticación y autorización a nivel de API.

Bases de Datos

Estructura y funcionalidad de PostgreSQL

Para realizar de forma satisfactoria la construcción de una base de datos que cumpla los requisitos del problema planteado y sea capaz de realizar solicitudes acordes con la información requerida para cada una de las funciones de los usuarios. Seguiremos una metodología bien definida.

Lo primero que realizaremos será crear un diagrama conceptual utilizando notación de Chen que nos dará una idea general de la estructura de la base. A continuación, tomaremos este diagrama y lo transformaremos a una representación relacional que muestre la estructura de las tablas que tendrá la base de datos final.

Modelo Conceptual

El primer paso por realizar para construir este diagrama es identificar las entidades fuertes, es decir las que poseen una llave distintiva para cada registro. Para nuestro caso todas las entidades utilizadas son de este tipo, pues una de las reglas implementadas es que entidades que de primera mano pueden ser consideradas como débiles (como solicitudes de camas) sean manejadas como un código distintivo para facilitar las transacciones.

Explicado lo anterior, podemos encontrar como encontrar entidades fuertes a *usuario*, *rol de usuario*, *patología*, *tipo de salón*, *salón*, *tipo de equipo médico*, *equipo médico reservación de cama*, *cama* y *procedimiento médico.* Identificados estas entidades procedemos a representarlos con sus atributos simples, compuestos y multivaluados.

Lo que seguiría a continuación es identificar las entidades débiles, pero no hay en nuestro caso. Ahora empezamos a realizar las relaciones entre entidades en donde identificamos las siguientes

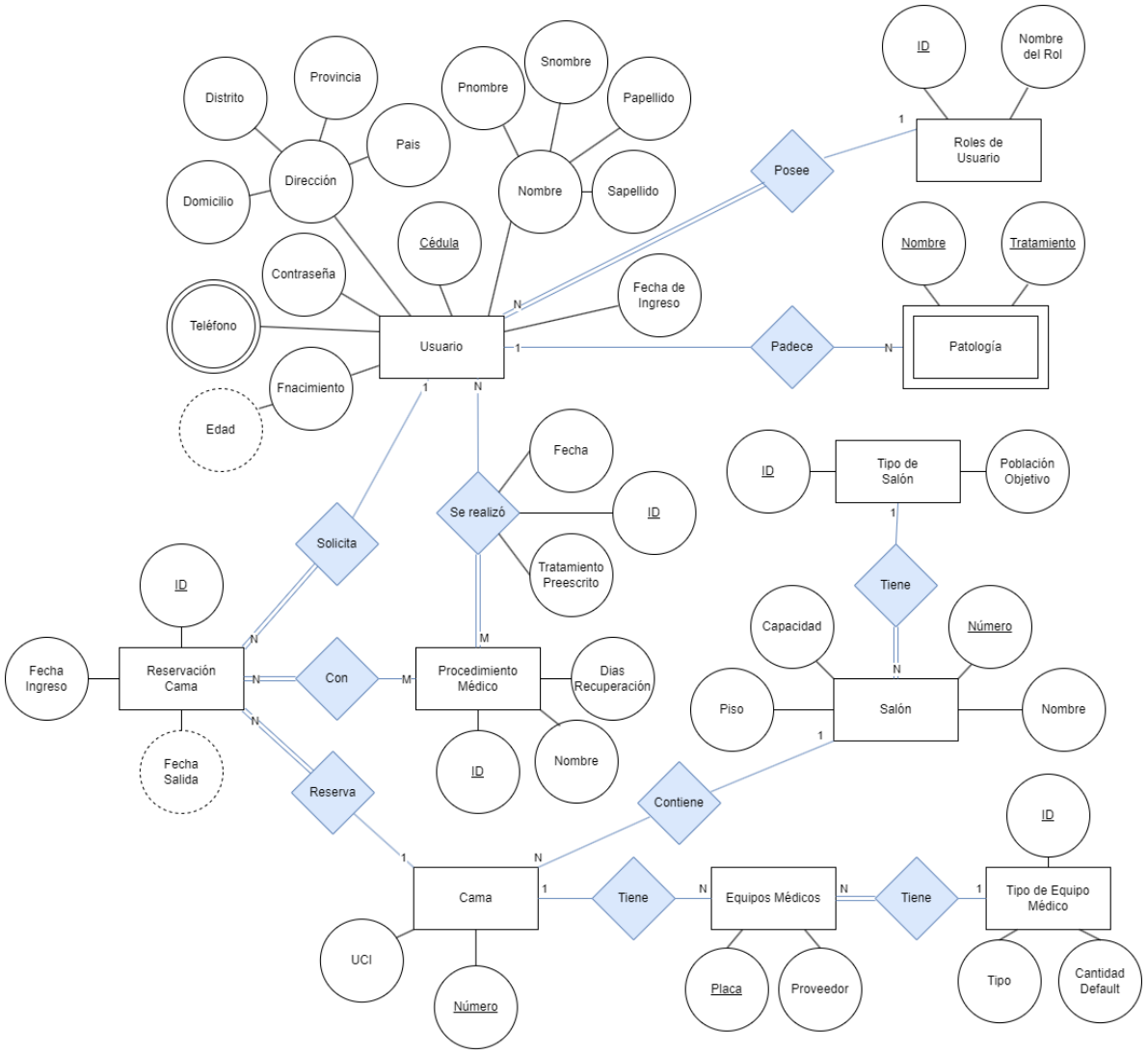


Figura 1. Diagrama conceptual del problema utilizando la notación de Chen.

Del diagrama anterior podemos identificar las siguientes relaciones:

Relación entre roles y usuarios: Cada usuario del sistema tiene un rol específico, como doctor, enfermero o administrador. Este rol define las funciones y permisos que el usuario tiene dentro del hospital.

Relación entre usuarios y números de teléfono: Un usuario puede tener varios números de teléfono registrados en el sistema, para poder ser contactado a través de cualquiera de ellos.

Relación entre usuarios y fechas de ingreso: Se lleva un registro de cada vez que un usuario ingresa al hospital, lo que permite saber su historial de visitas.

Relación entre usuarios y patologías: Cada usuario puede tener múltiples condiciones médicas o enfermedades registradas, junto con sus respectivos tratamientos.

Relación entre usuarios e historial médico: El historial médico de un usuario documenta todas las intervenciones y tratamientos que ha recibido, manteniendo un registro completo de su salud.

Relación entre procedimientos médicos e historial médico: Los procedimientos médicos realizados a un usuario son parte de su historial médico, documentando cada tratamiento específico que ha recibido.

Relación entre tipos de salón y salones: Los salones del hospital se clasifican en diferentes tipos, como salones para hombres, mujeres o niños, según su uso y características.

Relación entre salones y camas: Cada salón del hospital tiene varias camas donde los pacientes pueden ser acomodados durante su estancia.

Relación entre tipos de equipo y equipos médicos: Los equipos médicos del hospital se clasifican en distintos tipos, como desfibriladores, monitores o ultrasonidos, para facilitar su gestión y uso.

Relación entre camas y equipos médicos: Cada cama en el hospital puede estar equipada con varios dispositivos médicos necesarios para el cuidado del paciente.

Relación entre usuarios y reservaciones de camas: Los usuarios pueden reservar camas en el hospital para su estancia, asegurando que tengan un lugar disponible cuando lo necesiten.

Relación entre camas y reservaciones de camas: Una cama puede ser reservada por diferentes usuarios en distintas ocasiones, registrando cada uso de esta.

Relación entre procedimientos médicos y reservaciones de camas: Los procedimientos médicos específicos que un usuario necesita pueden estar asociados a su reservación de una cama en el hospital, garantizando que reciba el tratamiento adecuado durante su estancia.

Relación entre reservaciones de camas y procedimientos médicos: Cuando un usuario reserva una cama, también se especifican los procedimientos médicos que se le realizarán, asegurando una planificación adecuada de su tratamiento.

Modelo Relacional

Para realizar el modelo relacional nos basaremos en el modelo conceptual que construimos anteriormente, para ello seguiremos una serie de pasos específicos para que quede de la mejor forma posible.

1) Representar Entidades Fuertes

El primer paso es tomar las entidades fuertes, y representarlas con sus atributos, a excepción de los multivaluados. Como todas las entidades son fuertes: Rol de Usuario, Usuario, Procedimiento Médico, Tipo de Salón, Salón, Cama, Tipo de Equipo, Equipo Médico, Reservación de Cama, Historial Médico, Patología, Registro de las patologías que padecen los pacientes y sus respectivos tratamientos.

2) Graficar Entidades Débiles:

Al graficar las entidades fuertes nos dimos cuenta de que la fecha de ingreso del usuario era un atributo que iba a ser null en la mayoría de los casos, por lo tanto, lo agregamos como una entidad débil que deriva de usuario. Esta es la única entidad débil, por lo tanto, la representamos colocando la primary key de usuario dentro de esta tabla.

3) Representar Relaciones 1 a 1:

Para nuestro proyecto no se han identificado relaciones 1 a 1 en el modelo, ya que todas las relaciones entre las entidades permiten múltiples correspondencias.

4) Representar Relaciones 1 a N:

Usuario - Rol de Usuario: Cada usuario tiene un rol, pero un rol puede estar asignado a muchos usuarios. Esta relación se representa mediante una clave foránea (rol_id) en la tabla usuario que referencia la tabla rol_usuario.

Usuario - Historial Médico: Un usuario puede tener múltiples registros en su historial médico, pero cada registro pertenece a un solo usuario. Se usa una clave foránea (user_ced) en historial_medico que referencia la tabla usuario.

Usuario - Fecha de Ingreso: Un usuario puede ingresar al hospital múltiples veces, pero cada ingreso está asociado a un solo usuario. La relación se representa mediante una clave foránea (user_ced) en la tabla fecha_ingreso.

Usuario - Patología: Un usuario puede tener múltiples patologías registradas, pero cada patología está asociada a un solo usuario. Esto se maneja con una clave foránea (user_ced) en la tabla patologia.

Tipo de Salón - Salón: Un tipo de salón puede aplicarse a muchos salones, pero cada salón pertenece a un solo tipo. Esta relación se representa mediante una clave foránea (id_tipo) en la tabla salon.

Salón - Cama: Un salón puede tener múltiples camas, pero cada cama está asociada a un solo salón. Se usa una clave foránea (numero_salon) en la tabla cama que referencia la tabla salon.

Tipo de Equipo - Equipo Médico: Un tipo de equipo puede aplicarse a muchos equipos médicos, pero cada equipo médico pertenece a un solo tipo. Esta relación se representa mediante una clave foránea (id_tipo) en la tabla equipo_medico.

5) Representar Relaciones N a M:

Usuario - Reservación de Cama: Un usuario puede tener múltiples reservaciones de cama, y una cama puede ser reservada por diferentes usuarios en distintas ocasiones. Se representa con una tabla intermedia reservacion_cama que tiene claves foráneas (user_ced y num_cama) que referencian a las tablas usuario y cama.

Procedimiento Médico - Historial Médico: Un procedimiento médico puede estar en múltiples registros del historial médico, y un historial médico puede contener múltiples procedimientos. Esta relación se representa en la tabla historial_medico con claves foráneas (user_ced y id_procedimiento) que referencian a las tablas usuario y procedimiento_medico.

Procedimiento Médico - Reservación de Cama: Un procedimiento médico puede estar asociado a múltiples reservaciones de cama, y una reservación puede incluir múltiples procedimientos médicos. Se usa una tabla intermedia procedimiento_reservacion con claves

foráneas (id_procedimiento y id_reservacion) que referencian a las tablas procedimiento_medico y reservacion_cama.

6) Representar Atributos Multivaluados:

Teléfonos de Usuario: Un usuario puede tener varios números de teléfono. Esta relación se maneja con una tabla telefono_usuario, donde cada teléfono está asociado a un usuario mediante una clave foránea (user_ced) que referencia la tabla usuario.

El diagrama relacional de la base de datos se ve de la siguiente forma:

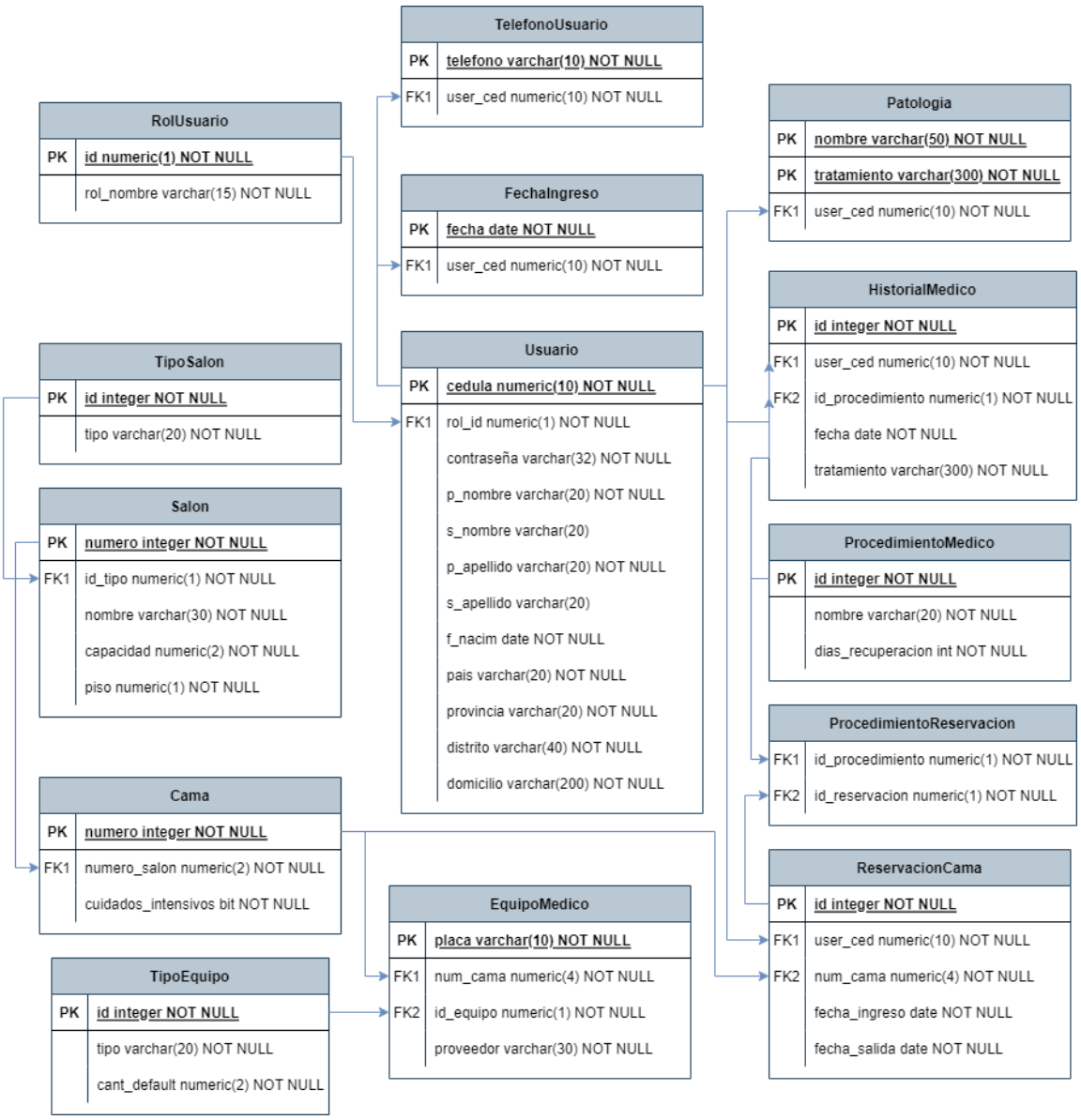


Figura 2. Diagrama Relacional de la base de datos implementada para HospiTEC

Para que el diseño de nuestra base de datos relacional sea eficiente y libre de redundancias se basa en las tres primeras formas normales (1FN, 2FN y 3FN). A continuación, se describe cómo se han implementado estas formas normales en nuestra base de datos, junto con las relaciones entre las entidades.

Primera Forma Normal (1FN)

La 1FN exige que los valores de cada columna de una tabla sean atómicos, es decir, no deben contener conjuntos o listas de valores.

En nuestra base de datos:

Las tablas están diseñadas de modo que cada columna contiene datos atómicos. Por ejemplo, en la tabla usuario, las columnas p_nombre, s_nombre, p_apellido, y s_apellido aseguran que cada parte del nombre del usuario se almacena en una columna separada.

Las tablas telefono_usuario, fecha_ingreso y patologia utilizan claves primarias compuestas para garantizar que cada entrada sea única y atómica.

Segunda Forma Normal (2FN)

La 2FN requiere que la base de datos esté en 1FN y que todas las columnas no clave dependan completamente de la clave primaria.

En nuestra base de datos:

En la tabla usuario, todas las columnas dependen completamente de la clave primaria cedula.

En las tablas telefono_usuario, fecha_ingreso y patologia, cada columna no clave depende completamente de la clave primaria compuesta.

Las tablas historial_medico y reservacion_cama también cumplen con esta forma normal, ya que todas sus columnas no clave dependen completamente de sus respectivas claves primarias.

Tercera Forma Normal (3FN)

La 3FN exige que la base de datos esté en 2FN y que no existan dependencias transitivas, es decir, cada columna no clave debe depender directamente de la clave primaria.

En nuestra base de datos:

La tabla usuario cumple con la 3FN, ya que columnas como pais, provincia y distrito dependen directamente de la clave primaria cedula y no de otras columnas.

La tabla equipo_medico asegura que columnas como proveedor dependen directamente de la clave primaria placa, eliminando dependencias transitivas.

La tabla procedimiento_reservacion establece relaciones directas entre procedimientos médicos y reservas de cama mediante sus claves foráneas id_procedimiento y id_reservacion.

Estructuras de Datos

Estructura de Tablas

A continuación, explicaremos de forma detallada las estructuras y constitución de las tablas utilizadas. Antes de iniciar tenemos que mencionar que la base usada para realizar la base de datos relacional de SQL y usamos como motor el entorno de desarrollo PostgreSQL.

Iniciamos con rol_usuario, esta tabla almacena los diferentes roles que pueden tener los usuarios en el sistema, como paciente, doctor, enfermero o administrador. Cada rol tiene un identificador único y un nombre descriptivo.

Pasando a usuario, la tabla contiene información personal y de contacto de los usuarios del sistema. Incluye datos como cédula, nombre, apellidos, fecha de nacimiento, dirección, y el rol asociado a cada usuario. También almacena la contraseña del usuario.

En telefono_usuario, esta tabla registra los números de teléfono asociados a cada usuario. Un usuario puede tener múltiples números de teléfono, y cada teléfono se asocia a la cédula del usuario.

En cuanto a fecha_ingreso, la tabla almacena los registros de las fechas en las que los usuarios ingresan al hospital. Cada registro incluye la cédula del usuario y la fecha del ingreso.

La tabla patologia registra las patologías que padecen los pacientes, así como los tratamientos asociados. Cada patología se asocia a la cédula del usuario que la padece.

Hablando de procedimiento_medico, esta tabla contiene información sobre los diferentes procedimientos médicos disponibles en el hospital. Cada procedimiento tiene un identificador único, un nombre y el número de días necesarios para la recuperación.

En historial_medico, la tabla almacena los registros de los procedimientos médicos realizados a los pacientes. Incluye información como la fecha del procedimiento, el tratamiento prescrito, la cédula del usuario y el identificador del procedimiento realizado.

La tabla tipo_salon categoriza los diferentes tipos de salones disponibles en el hospital. Cada tipo de salón tiene un identificador único y un nombre descriptivo.

Pasando a salon, la tabla almacena información sobre los salones del hospital, incluyendo su número, nombre, capacidad en camas, el piso en que se encuentran y el tipo de salón al que pertenecen.

En cama, esta tabla registra las camas disponibles en los salones del hospital. Cada cama tiene un número único, está asociada a un salón específico y se indica si es una cama de cuidados intensivos.

La tabla tipo_equipo clasifica los diferentes equipos médicos disponibles en el hospital. Cada tipo de equipo tiene un identificador único, un nombre descriptivo y la cantidad por defecto de ese tipo de equipo.

En equipo_medico, esta tabla almacena información sobre los equipos médicos, incluyendo su placa, el tipo de equipo, el proveedor y la cama a la que está asignado.

Pasando a reservacion_cama, la tabla registra las reservas de camas hechas por los pacientes. Cada reservación incluye la cédula del usuario, el número de la cama reservada, la fecha de ingreso y la fecha de salida.

Finalmente, procedimiento_reservacion relaciona los procedimientos médicos con las reservaciones de cama. Cada registro indica qué procedimiento médico se asocia con una determinada reservación de cama.

Estructura de Procedimientos almacenados

Ahora procederemos a hablar de la lógica de las solicitudes a la base de datos, para ellos iniciaremos con los procedimientos almacenados utilizados.

Insertar un nuevo usuario: Este procedimiento añade un nuevo usuario al sistema, registrando toda su información personal y de contacto, incluyendo el rol que desempeña, su cédula, nombre, apellidos, fecha de nacimiento, dirección y contraseña.

1) Modificar la contraseña de un usuario: Permite actualizar la contraseña de un usuario específico en el sistema.
2) Insertar un nuevo salón: Este procedimiento añade un nuevo salón a la base de datos, registrando su tipo, nombre, capacidad y piso.
3) Modificar un salón existente: Permite actualizar la información de un salón específico, incluyendo su tipo, nombre, capacidad y piso.
4) Eliminar un salón: Este procedimiento elimina un salón específico de la base de datos.
5) Insertar un nuevo equipo médico: Añade un nuevo equipo médico al sistema, registrando su placa, tipo, número de cama al que está asignado y el proveedor.
6) Modificar un equipo médico existente: Permite actualizar la información de un equipo médico específico, como su tipo, número de cama y proveedor.
7) Eliminar un equipo médico: Este procedimiento elimina un equipo médico específico de la base de datos utilizando su placa.
8) Insertar una nueva cama: Añade una nueva cama a la base de datos, registrando el número de salón al que pertenece y si es de cuidados intensivos.
9) Modificar una cama existente: Permite actualizar la información de una cama específica, incluyendo el número de salón y si es de cuidados intensivos.
10) Eliminar una cama: Este procedimiento elimina una cama específica de la base de datos.
11) Insertar un nuevo procedimiento médico: Añade un nuevo procedimiento médico a la base de datos, registrando su nombre y los días de recuperación necesarios

12) Modificar un procedimiento médico existente: Permite actualizar la información de un procedimiento médico específico, como su nombre y los días de recuperación.
13) Eliminar un procedimiento médico: Este procedimiento elimina un procedimiento médico específico de la base de datos.
14) Modificar la información del personal médico: Actualiza la información personal y de contacto de un miembro del personal médico, así como su rol y fecha de ingreso al sistema.
15) Eliminar un miembro del personal médico: Este procedimiento elimina un miembro del personal médico de la base de datos utilizando su cédula.
16) Insertar una nueva patología: Este procedimiento añade una nueva patología al sistema, registrando su nombre, el tratamiento asociado y la cédula del usuario afectado.
17) Ingresar un nuevo historial médico: Permite registrar una nueva entrada en el historial médico de un usuario, incluyendo la fecha del procedimiento, el tratamiento realizado, la cédula del usuario y el identificador del procedimiento médico.
18) Editar un historial médico: Este procedimiento permite actualizar la información de una entrada existente en el historial médico, como la fecha del procedimiento, el tratamiento realizado, la cédula del usuario y el identificador del procedimiento.
19) Insertar una relación entre procedimiento y reservación: Añade una nueva relación entre un procedimiento médico y una reservación de cama, registrando los identificadores de ambos para vincularlos correctamente en el sistema.

Estructura de Funciones

Ahora vamos a hablar de la estructura de las funciones utilizadas. En su mayoría fueron utilizadas para obtener valores, tablas en información de la tabla.

1. Verificar Inicio de Sesión: Esta función verifica las credenciales de inicio de sesión de un usuario, validando si la cédula y la contraseña proporcionadas coinciden con los registros en la base de datos.
2. Obtener Información de Usuario: Permite obtener información detallada de un usuario específico a partir de su cédula, incluyendo su nombre, apellidos, edad, país, provincia, distrito, domicilio y rol asociado en el sistema.

3. Obtener Salones: Retorna una tabla con información sobre todos los salones disponibles en el sistema, incluyendo su número, tipo, nombre, capacidad y piso.
4. Obtener Camas: Esta función devuelve detalles sobre todas las camas disponibles en el sistema, mostrando su número, el número del salón al que pertenecen, si son de cuidados intensivos y los tipos de equipos médicos asociados.
5. Obtener Procedimientos: Retorna una tabla con información sobre todos los procedimientos médicos disponibles en el sistema, incluyendo su identificador, nombre y días de recuperación.
6. Obtener Personal Médico: Permite obtener información detallada sobre el personal médico del sistema, incluyendo su cédula, nombre, apellidos, fecha de nacimiento, edad, fecha de ingreso, país, provincia, distrito y domicilio.
7. Insertar una nueva patología: Este procedimiento añade una nueva patología al sistema, registrando su nombre, el tratamiento asociado y la cédula del usuario afectado.
8. Ingresar un nuevo historial médico: Permite registrar una nueva entrada en el historial médico de un usuario, incluyendo la fecha del procedimiento, el tratamiento realizado, la cédula del usuario y el identificador del procedimiento médico.
9. Editar un historial médico: Este procedimiento permite actualizar la información de una entrada existente en el historial médico, como la fecha del procedimiento, el tratamiento realizado, la cédula del usuario y el identificador del procedimiento.
10. Insertar una relación entre procedimiento y reservación: Añade una nueva relación entre un procedimiento médico y una reservación de cama, registrando los identificadores de ambos para vincularlos correctamente en el sistema.

Estructura de Triggers

También utilizamos un par de triggers para modificar algunos valores a la hora de insertar valores en la base de datos, el resumen de esta funcionalidad lo vemos a continuación.

Triggers de incremento de cantidad disponible: Se ha creado una función fn_incrementar_cant_disponible() que se activará después de cada inserción en la tabla equipo_medico. Esta función actualiza la cantidad predeterminada de un tipo de equipo en la tabla tipo_equipo, aumentándola en 1. El trigger asegura que cada vez que se agregue un

nuevo equipo médico, la cantidad disponible de ese tipo de equipo se incremente automáticamente.

Triggers de inserción de fecha de ingreso: Se ha creado una función fn_insertar_fecha_ingreso() que se activará después de cada inserción en la tabla usuario cuando el rol asociado sea un médico o una enfermera (rol_id 2 o 3). Esta función inserta la fecha de ingreso actual junto con la cédula del usuario en la tabla fecha_ingreso. El trigger garantiza que se registre automáticamente la fecha de ingreso de los nuevos miembros del personal médico en la base de datos.

Estructura y funcionalidad de Mongo DB

Para utilizar MongoDB a partir de los servicios en la nube, se utiliza Azure. En Azure, para utilizar MongoDB como servicio en la nube, se debe crear un clúster de Azure Cosmos DB para MongoDB. Este proceso comienza con la creación de un Resource Group, que es un contenedor lógico en Azure que agrupa recursos relacionados. Al crear un Resource Group, se especifica que el único recurso inicial que se utilizará es "Azure Cosmos DB for MongoDB".

Una vez creado el Resource Group, se procede a crear el clúster de Azure Cosmos DB para MongoDB. Durante la creación del clúster, se configuran aspectos importantes como la región geográfica donde se alojará, el nivel de rendimiento deseado, las opciones de red y la seguridad. En cuanto a la configuración de red, se puede agregar la dirección IP 0.0.0.0, lo que permite acceder a la base de datos desde cualquier dirección IP.

Además, durante la creación del clúster, se requiere establecer un usuario y una contraseña que se utilizarán para acceder a la instancia de MongoDB. Una vez que el clúster esté creado, Azure generará una cadena de conexión (connection string) que contiene información crucial como el nombre de usuario, la contraseña, el nombre del clúster y otras configuraciones necesarias para establecer una conexión segura.

Esta cadena de conexión es el punto de entrada para que las aplicaciones y herramientas de cliente de MongoDB, como MongoDB Compass, puedan interactuar con la base de datos MongoDB alojada en Azure Cosmos DB. Al ingresar la cadena de conexión en una herramienta como MongoDB Compass, se establece una conexión segura con la instancia de MongoDB en la nube de Azure.

A partir de este momento, las aplicaciones pueden realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en las bases de datos y colecciones de MongoDB, de la misma manera que lo harían con una instancia local de MongoDB. La ventaja es que Azure Cosmos DB para MongoDB se encarga automáticamente de tareas complejas como el escalado, la replicación, la alta disponibilidad y el particionamiento de datos, lo que facilita la gestión y el mantenimiento de la base de datos en la nube.

Utilizando MongoDB Compass, se ingresa el connection string para conectarse al clúster y aquí se procede a crear la base de datos llamada HospiTEC. Dentro de esta base, se crea la colección evalServicio, donde se guardarán las evaluaciones de servicio realizadas por los pacientes.

Para la implementación de esto en la aplicación web, dentro de la carpeta src/components/MongoDB se crean funciones para conectarse a la base utilizando el ya mencionado connection string. Una función para insertar una nueva evaluación, cuya inserción requiere de la cédula del paciente, la calificación dada en las áreas de aseo, trato y puntualidad, además de una entrada para comentarios adicionales que desee agregar el paciente. También se crea una función para obtener todas las evaluaciones presentes en la base de datos. Ambas funciones son exportables y, por lo tanto, solo es cuestión de llamarlas para realizar cualquiera de estas acciones.

API / Web Service / Rest Service

Para la API, se utiliza C# / .NET Core para su programación, y para su testeo, se utiliza la herramienta Swagger enviando jsons o información en el mismo URL, dependiendo de la cantidad de información que el procedimiento de la base de datos necesita.

Primero, en la carpeta Controllers se encuentra toda la lógica principal para manejar los procedimientos y funciones que tiene la base de datos, seguidamente, en la carpeta Models, se encuentran los modelos base que se utiliza, ya sea por los controladores que se conectan a la base de datos de PostgreSQL o a la base de datos de MongoDB. En la carpeta Class se encuentran las clases bases o generales de las cuales derivan el resto de las clases que usan los controladores. Finalmente, en Program.cs se encuentra la configuración a las conexiones de la base de datos de PostgreSQL o MongoDB, utilizando el string de conexión que se encuentra en appsettings.json, ya sea "ConnectionString / DefaultConnextion" para la base de datos de PostgreSQL o "MongoDBSettings / ConnectionString, DatabaseName, CollectionName" para la base de datos de MongoDB.

La estructura base de cada controlador es la misma para todos, primero, todos los controladores son clases los cuales están conectados a una ruta http, y para diferenciarlos, se usa el nombre de la clase como parte de la ruta http. Y gracias al modelo creado para cada base de datos, este hereda la información de conexión a cada controlador para que estos logren realizar los llamados necesarios.

Es importante recalcar que, se cuenta con pocos archivos de formato .cs para los controladores, ya que varios envían o reciben información muy similar a otros controladores, por lo que hay varios llamados a funciones o procedimiento en un mismo documento, para así mantener el orden y reutilizar clases.

Aplicación Web

Lo aplicación web es donde tanto doctores, administradores, enfermeros y pacientes , hacen sus funcionalidades principales. En la **vista de administradores**, estos pueden:

- Gestionar la información de las camas
- Gestionar la información de los salones
- Gestionar la información de los procedimientos médicos
- Gestionar la información del equipo médico
- Ver la información de los reportes generados por los pacientes

Los Doctores y enfermeros pueden:

- Registrar pacientes nuevos conforme van llegando a las consultas.
- Agregar patologías de los pacientes conforme sea necesario.
- Ver el historial clínico de los pacientes, además de poder editarlo y agregar alguno.

Finalmente, **la vista de pacientes puede**:

- Ver las camas, su horario, y hacer las solicitudes de camas necesarias.
- Observar las reservaciones que tiene ya registradas.
- Ver historiales médicos que ya tiene registrados, ordenados de fecha más antigua a fecha más nueva.
- Evaluar la estadía que tuvo el paciente, pudiendo evaluar el aseo del hospital, trato del personal y puntualidad de las citas. Finalmente puede agregar un comentario.

Componentes Principales:

AdminPage: encargada de toda la lógica de la vista de administrador antes mencionada. **LoginPage:** encargada del logueo del usuario.

PacientePage: encargada de todas las funcionalidades de la vista de paciente. **RegisterPage:** encargada de registrar a operadores.

CamaSol: encargada del formulario para reservar el laboratorio por parte de un paciente. Calendar**:** encargada de observar todas las reservas de x laboratorio.

Y finalmente

DoctorPage: encargada de toda la funcionalidad de la vista de profesor. Se tiene un css principal, para tener toda la lógica necesaria ahí, pero cada vista tiene su css, para cambios específicos de la página.

Cabe mencionar la importancia de **assets**, que contiene todas las imágenes y anexos necesarios para que corra la aplicación.

Problemas resueltos:

Para la base de datos:

- Uno de los principales problemas fue el manejo de los choques entre citas, pues generar resulto de gran dificultad generar una función que lo verificara correctamente. Para solucionarlo se investigó e implementó la función “Between” para manejar colisiones entre rangos de tiempo.

Para la página web:

- No se han encontrado problemas en la página web

Para la API:

- No se han encontrado problemas en la API

Problemas conocidos:

Para la base de datos:

- No se puede realizar la eliminación de personal, equipo, salones y citas pues están relacionados con otras tablas de la base, es decir que existan como llaves foráneas en otras tablas. Esto es una característica asociada a las bases de datos relacionales. Para solucionarlo se debe evitar de eliminar registros o editar sus referencias.

Para la página web:

- No se han encontrado problemas con la página web

Para la API:

- No se han encontrado problemas con la API

Evidencias de trabajo:

Al realizar previamente un proyecto similar como tarea del curso, los roles del equipo se mantuvieron y la división de trabajos fue sencilla de realizar, pues, la mayoría trabajo en lo que había desarrollado previamente.

La comunicación de los miembros de trabajo se realizó a través de la herramienta de WhatsApp. En un grupo destinado a las tarea del proyecto. Aquí se informó acerca de los avances de cada estudiante, el estado de cada uno de los componentes y la coordinación para trabajar en unísono. Ejemplo de ello podemos observar a continuación:

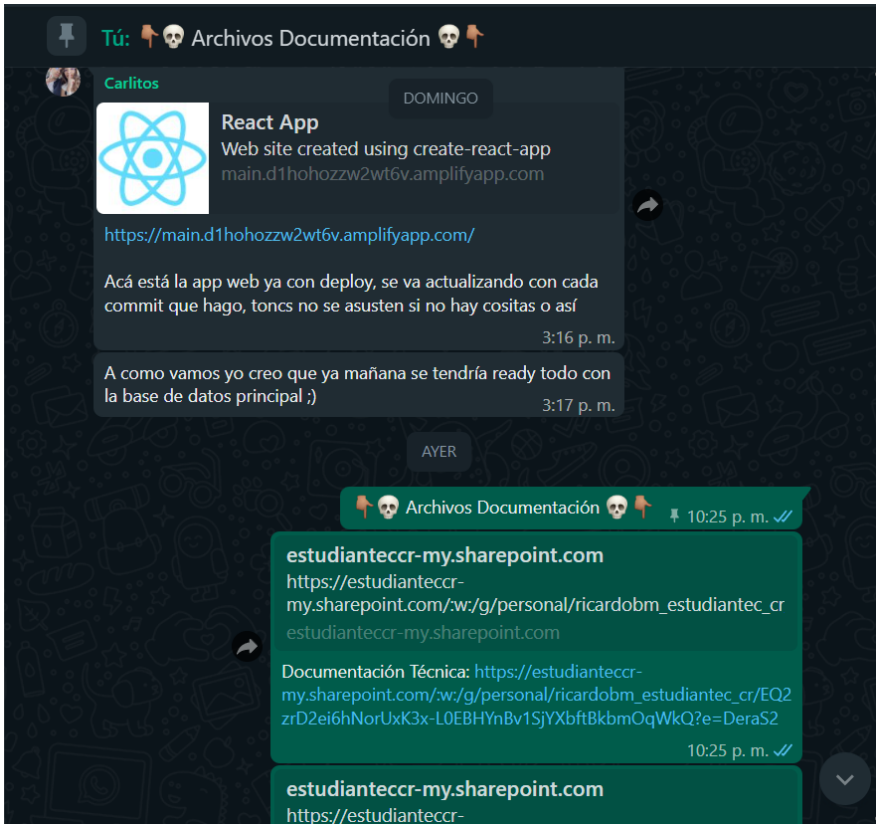


Figura 3. Evidencia del canal de comunicación principal del equipo de trabajo (WhatsApp)

Además, se utiliza la herramienta de versiones de código GitHub, de esta forma podíamos trabajar de forma cómoda en cada una de las tareas, consultar el avance de las demás personas y gestionar la unión de los componentes evitando cualquier conflicto de compatibilidad que pudiera surgir en caso de no utilizar controladores de versiones.

A continuación, podemos observar la división de ramas del repositorio, con la finalidad de organizar las secciones del proyecto y modularizar lo mayor posible los componentes garantizando así una alta escalabilidad y un menor acoplamiento:

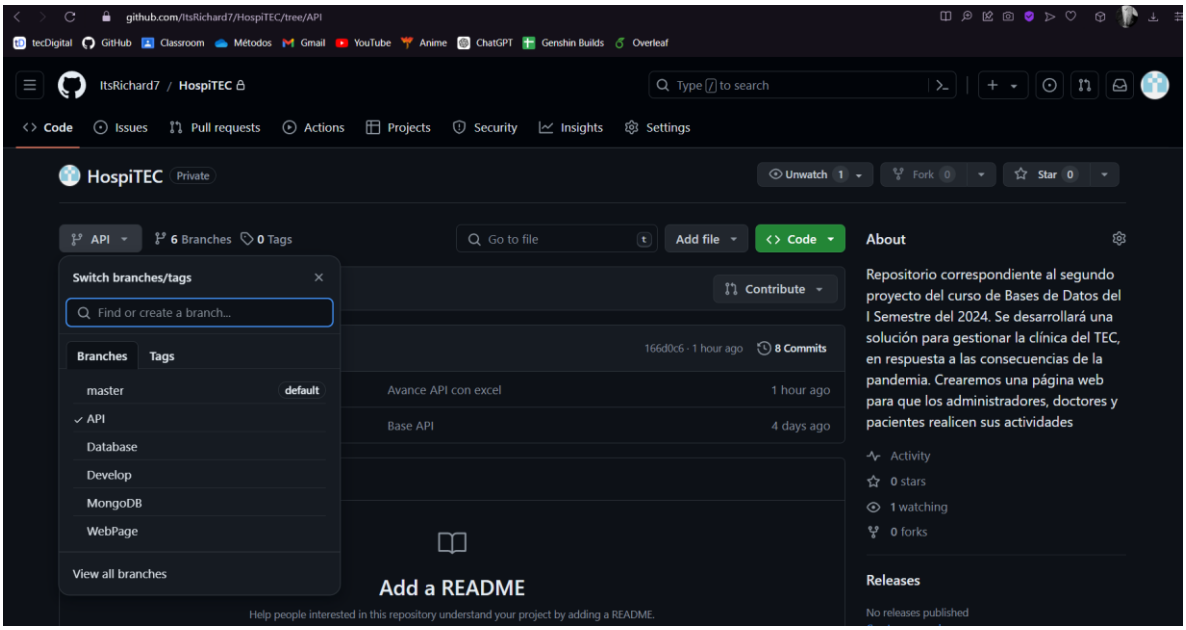
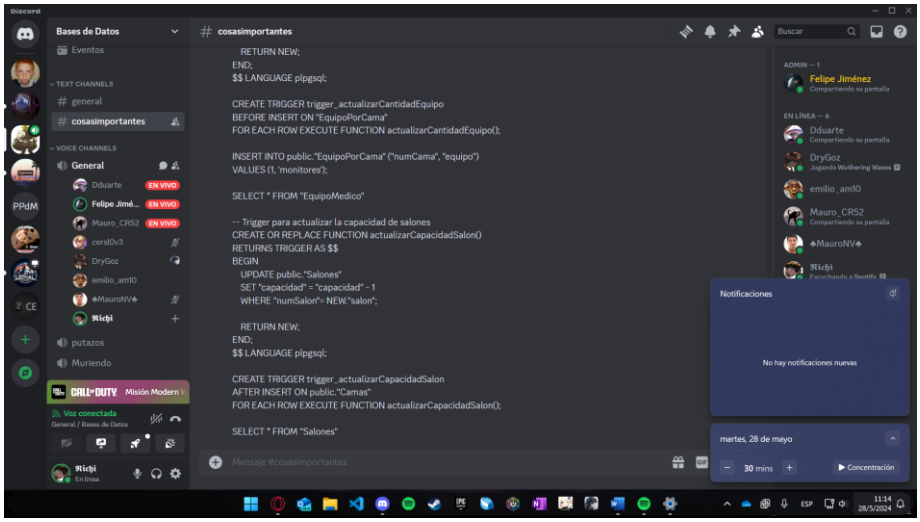
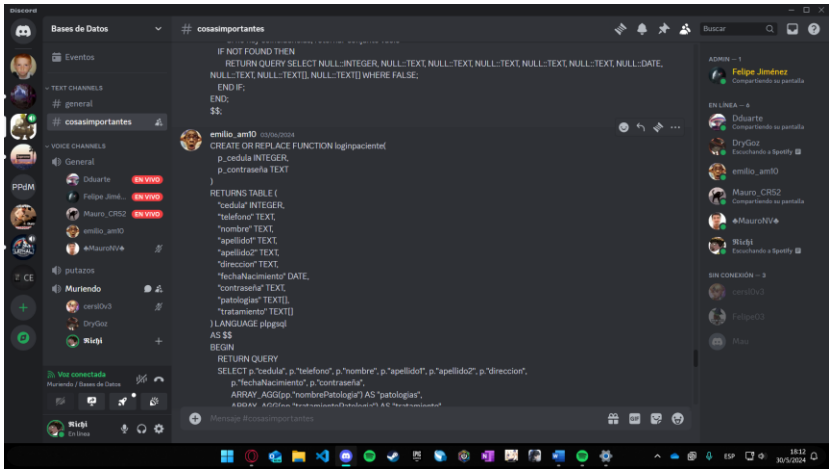
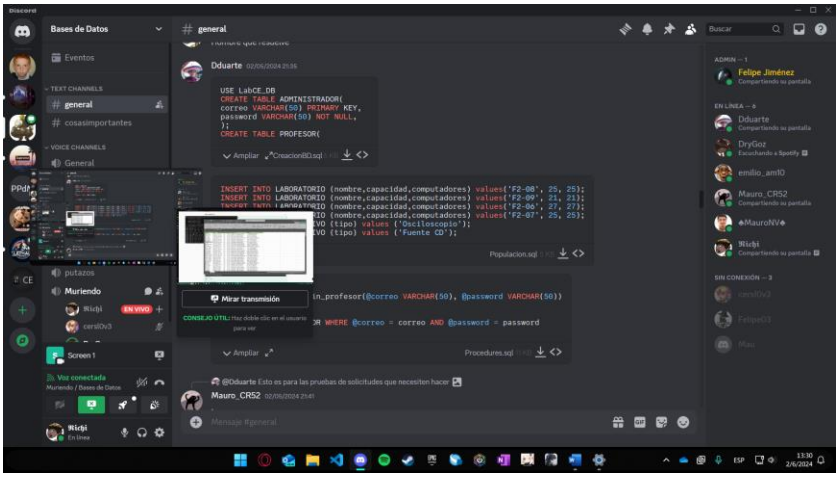


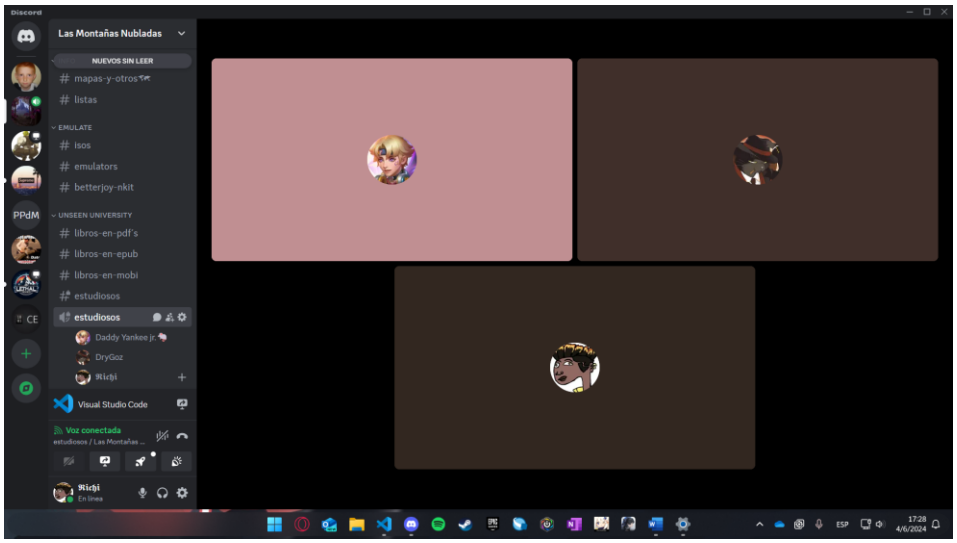
Figura 4. Repositorio de GitHub del proyecto y sus ramas correspondientes a las funcionalidades desarrolladas.

Ahora a continuación, podemos observar algunas minutas de reuniones de trabajos hechas por el equipo de trabajo para desarrollar todas las funcionalidades de la aplicación.


Discord
Bases de Datos
Eventos
TEXT CHANNELS
general
cosasimportantes
VOICE CHANNELS
General
Dduarte
Felipe Jimé...
Mauro_CR52
cersl0v3
DryGoz
emilio_am10
♣MauroNV♣
Richi
putazos
Muriendo
CALL OF DUTY Misión Modern
Voz conectada
General / Bases de Datos
Richi
En línea
cosasimportantes
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_actualizarCantidadEquipo
BEFORE INSERT ON "EquipoPorCama"
FOR EACH ROW EXECUTE FUNCTION actualizarCantidadEquipo();
INSERT INTO public."EquipoPorCama" ("numCama", "equipo")
VALUES (1, 'monitores');
SELECT * FROM "EquipoMedico"
-- Trigger para actualizar la capacidad de salones
CREATE OR REPLACE FUNCTION actualizarCapacidadSalon()
RETURNS TRIGGER AS $$
BEGIN
UPDATE public."Salones"
SET "capacidad" = "capacidad" - 1
WHERE "numSalon" = NEW."salon";
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_actualizarCapacidadSalon
AFTER INSERT ON public."Camas"
FOR EACH ROW EXECUTE FUNCTION actualizarCapacidadSalon();
SELECT * FROM "Salones"
Mensaje #cosasimportantes
Buscar
ADMIN — 1
Felipe Jiménez
Compartiendo su pantalla
EN LÍNEA — 6
Dduarte
Compartiendo su pantalla
DryGoz
Jugando Wuthering Waves
emilio_am10
Mauro_CR52
Compartiendo su pantalla
♣MauroNV♣
Richi
Notificaciones
No hay notificaciones nuevas
martes, 28 de mayo
30 mins
Concentración
ESP
11:14
28/5/2024


Discord
Bases de Datos
Eventos
TEXT CHANNELS
general
cosasimportantes
VOICE CHANNELS
General
Dduarte
Felipe_Jimé...
Mauro_CR52
emilio_am10
♣MauroNV♣
putazos
Muriendo
cersl0v3
DryGoz
Richi
Voz conectada
Muriendo / Bases de Datos
Richi
En línea
cosasimportantes
Buscar
IF NOT FOUND THEN
RETURN QUERY SELECT NULL::INTEGER, NULL::TEXT, NULL::TEXT, NULL::TEXT, NULL::TEXT, NULL::TEXT, NULL::DATE, NULL::TEXT, NULL::TEXT[], NULL::TEXT[] WHERE FALSE;
END IF;
END;
$$;
emilio_am10 03/04/2024
CREATE OR REPLACE FUNCTION loginpaciente(
p_cedula INTEGER,
p_contraseña TEXT
)
RETURNS TABLE (
"cedula" INTEGER,
"telefono" TEXT,
"nombre" TEXT,
"apellido1" TEXT,
"apellido2" TEXT,
"direccion" TEXT,
"fechaNacimiento" DATE,
"contraseña" TEXT,
"patologias" TEXT[],
"tratamiento" TEXT[]
) LANGUAGE plpgsql
AS $$
BEGIN
RETURN QUERY
SELECT p."cedula", p."telefono", p."nombre", p."apellido1", p."apellido2", p."direccion",
p."fechaNacimiento", p."contraseña",
ARRAY_AGG(pp."nombrePatologia") AS "patologias",
Mensaje #cosasimportantes
ADMIN — 1
Felipe Jiménez
Compartiendo su pantalla
EN LÍNEA — 6
Dduarte
Compartiendo su pantalla
DryGoz
Escuchando a Spotify
emilio_am10
Mauro_CR52
Compartiendo su pantalla
♣MauroNV♣
Richi
Escuchando a Spotify
SIN CONEXIÓN — 3
cersl0v3
Felipe03
Mau
ESP
18:12
30/5/2024


Bases de Datos
general
Eventos
TEXT CHANNELS
general
cosasimportantes
VOICE CHANNELS
General
putazos
Muriendo
Screen 1
Voz conectada
Dduarte
USE LabCE_DB
CREATE TABLE ADMINISTRADOR(
correo VARCHAR(50) PRIMARY KEY,
password VARCHAR(50) NOT NULL,
);
CREATE TABLE PROFESOR(
Ampliar
INSERT INTO LABORATORIO (nombre,capacidad,computadores) values('F2-08', 25, 25);
INSERT INTO LABORATORIO (nombre,capacidad,computadores) values('F2-09', 21, 21);
INSERT INTO LABORATORIO (nombre,capacidad,computadores) values('F2-06', 27, 27);
INSERT INTO LABORATORIO (nombre,capacidad,computadores) values('F2-07', 25, 25);
INSERT INTO TIPO (tipo) values ('Osciloscopio');
INSERT INTO TIPO (tipo) values ('Fuente CD');
Mirar transmisión
Ampliar
@Oduarte Esto es para las pruebas de solicitudes que necesiten hacer
Mauro_CR52
Mensaje #general
Buscar
Felipe Jiménez
Dduarte
DryGoz
emilio_aml0
Mauro_CR52
Richi
ESP


Discord
Las Montañas Nubladas
NUEVOS SIN LEER
mapas-y-otros
listas
EMULATE
isos
emulators
betterjoy-nkit
UNSEEN UNIVERSITY
libros-en-pdf's
libros-en-epub
libros-en-mobi
estudiosos
estudiosos
Daddy Yankee jr.
DryGoz
Richi
Visual Studio Code
Voz conectada
estudiosos / Las Montañas ...
Richi
En línea
ESP
17:28
4/6/2024

Conclusiones

Se implementaron los conocimientos de diagramas conceptuales y relacionales para desplegar correctamente «usando AWS» una base de datos en PostgreSQL, que se adecuará a las necesidades del hospital, como pudimos observar en la sección de la estructura y funcionamiento de las bases de datos usadas.

Se logró desplegar satisfactoriamente «usando Azure» un API/WebService/REST Service como podemos observar en la sección llamada de esta forma. Así logramos facilitar la transmisión de información y comunicación entre la aplicación web, la aplicación móvil y la base de datos.

Se desarrolló y desplegó «usando AWS» de forma exitosa una página web para administradores, médicos y pacientes utilizando la herramienta de React y sus componentes característicos, evidencia de ello pudimos observar en las descripciones de todas las partes que conforman el sitio web en la sección “Página Web” .

Por último, se logró cumplir totalmente las especificaciones del producto pues gracias a todos los productos mencionados anteriormente se desarrolló un prototipo de aplicación que permite usuarios de los laboratorios de ingeniería de computadores realizar sus actividades de manera efectiva, rápida, confiable y centralizada TEC.

Recomendaciones

- Antes de comenzar a codificar, es recomendable tener un plan claro. Esto debería incluir una comprensión de las necesidades del proyecto, así como una idea de las estructuras de datos y las funciones que se utilizarán.
- Es crucial entender cómo se estructuran los objetos JSON y cómo se pueden manipular. Esto permitirá un manejo más eficiente de estos objetos y evitará errores comunes.
- Antes de utilizar una REST API, es recomendable definir primero los modelos que se van a utilizar. Esto puede ayudar a evitar la creación de modelos innecesarios y a mantener el código más limpio y eficiente.
- Antes de diseñar la base de datos, es recomendable realizar diagramas que funcionen como guía y modelo del proyecto. En nuestro caso utilizamos diagramas conceptuales y relacionales, pero no tiene por qué limitarse solo a estos dos.

Bibliografía

Tutorial: Intro to React – React. (s. f.). React. https://legacy.reactjs.org/tutorial/tutorial.html

Learn the Basics · React Native. (2023, 8 diciembre). https://reactnative.dev/docs/tutorial

C# Tutorial (C Sharp). (s. f.). https://www.w3schools.com/cs/index.php

Jc Miron (Director). (2022, Julio 4). *SETUP C#, .Net Core, and VS Code in 2 MINS! 2022*. https://www.youtube.com/watch?v=SQim2adwVJI

Rodrigo Méndez (Director). (2023, septiembre 20). *CURSO Básico de REACT NATIVE*. https://www.youtube.com/watch?v=XlraT4cAS9E

Programming with Mosh. (2023, March 12). *React tutorial for beginners* [Video]. YouTube. https://www.youtube.com/watch?v=SqcY0GlETPk

Fireship. (2020, September 8). *React in 100 seconds* [Video]. YouTube. https://www.youtube.com/watch?v=Tn6-PIqc4UM

Datatable in React JS. (n.d.). YouTube. https://www.youtube.com/playlist?list=PLZboEx3gZXge7zhg45b5TzmjhNPudSaZn

Code With Yousaf. (2023, February 6). *React Axios Crud App with JSON server | ReactJS Axios Rest API | React Crud App with JSON server* [Video]. YouTube. https://www.youtube.com/watch?v=Rm4__WgPncI

Elmasri y Navathe. Fundamentals of Database Systems, 6ta Edición. Addison Wesley. 2010.

Raheem, Nasir. Big Data: A Tutorial-Based Approach, 2019

Albert Y Zomaya editor.; Sherif Sakr. Handbook of Big Data Technologies. SpringerLink. 2017

M. Tamer Özsu Patrick Valduriez. Principles of distributed database systems. 2011

Wiese, Lena. Advanced Data Management: For SQL, NoSQL, Cloud and Distributed Databases. 2015