

UNIVERSIDAD TÉCNICA DE AMBATO

**FACULTAD DE: INGENIERÍA EN SISTEMAS,
ELECTRÓNICA E INDUSTRIAL**

CARRERA DE: Tecnologías de la Información

SISTEMAS DE BASES DE DATOS DISTRIBUIDOS

DOCENTE:

JOSE RUBEN CAIZABA CAIZABUANO

AGOSTO 2025 - ENERO 2026



INFORME DE PROYECTO PRIMER PARCIAL

I. PORTADA

Tema:	Arquitectura MVC y Bases de Datos Distribuidas Avanzadas
Unidad de Organización Curricular:	PROFESIONAL
Nivel y Paralelo:	Quinto "A"
Alumnos participantes:	Aldas Jordan Wellington Ismael Caguasango Bayas Alex Patricio Gómez Llerena Luis Fernando Paredes Barrera Luis Enrique
Asignatura:	Sistemas de Base de Datos Distribuidas
Docente:	Ing. Jose Ruben Caiza Caizabuano, Mg.

II. INFORME DE GUÍA PRÁCTICA

2.1 Objetivos

2.1.1 Objetivo General:

Desarrollar una aplicación distribuida que utilice la arquitectura MVC para gestionar la información de un sistema hospitalario integrado por múltiples centros médicos. La solución debe exponer servicios mediante API RESTful, garantizar la replicación y sincronización de datos en bases de datos distribuidas y enfrentar desafíos avanzados como la selección del modelo de distribución, configuración en contenedores y esquemas de replicación.

2.1.2 Objetivos Específicos:

- Implementar un sistema hospitalario distribuido en PostgreSQL, aplicando fragmentación horizontal para distribuir los datos de consultas y pacientes entre los diferentes centros médicos (Quito, Guayaquil y Cuenca), garantizando una administración eficiente y descentralizada de la información.
- Desarrollar la capa backend bajo la arquitectura MVC, utilizando Node.js y Express para estructurar los modelos, controladores y vistas (respuestas JSON), permitiendo la comunicación fluida entre la base de datos distribuida y los frontends desarrollados en Vue.js y React.
- Configurar un esquema de replicación pasiva entre las bases de datos, donde el nodo central (Quito) actúe como maestro y los nodos secundarios (Guayaquil y Cuenca) como réplicas de acceso, evaluando el uso de replicación sincrónica o asincrónica según la criticidad de los datos y la eficiencia del sistema.

2.2 Modalidad

Presencial

2.3 Tiempo de duración

Presenciales: 6

No presenciales: 0



2.4 Instrucciones

Parte 1: Configuración de la Infraestructura y Bases de Datos Avanzadas

Infraestructura en la Nube (Azure): Crear tres Máquinas Virtuales (VMs) (Quito como central; Guayaquil y Cuenca como locales). Configurar redes seguras (VNet y NSG).

Bases de Datos Distribuidas: Instalar un SGBD (MariaDB/MySQL/PostgreSQL) en cada VM.

Modelo de Distribución: Implementar un esquema Híbrido que combine:

- Modelo Centralizado/Replicado para datos globales (Médicos, Empleados).
- Fragmentación Horizontal para las Consultas Médicas (separar por centro).
- Fragmentación Vertical para datos sensibles (opcional, pero recomendado para alcanzar el nivel avanzado de "Fragmentación Híbrida").

Esquema de Replicación: Configurar la replicación (ej. Pasiva/Maestro-Esclavo Asincrónica) para sincronizar datos globales entre nodos.

Docker: Configurar al menos un nodo distribuido usando contenedores Docker como alternativa o complemento a las VMs.

Acceso Remoto: Garantizar la conectividad remota segura entre los nodos de la base de datos.

Parte 2: Desarrollo del Backend con Arquitectura MVC y API RESTful

Arquitectura MVC: Desarrollar el backend con Node.js/Express siguiendo el patrón

Modelo-Vista-Controlador.

- Modelo: Debe gestionar la lógica de distribución de datos, dirigiendo lecturas a nodos replicados y escrituras de consultas a nodos locales fragmentados.
- Controlador: Definir los endpoints API RESTful para las operaciones CRUD sobre todas las entidades (Centros, Médicos, Especialidades, Empleados, Consultas).
- Vista: Generar respuestas estandarizadas en formato JSON. Documentación API: Documentar todos los endpoints utilizando herramientas como Swagger.

Parte 3: Desarrollo del Frontend e Interfaces de Usuario

Interfaz de Administración (Vue.js): Desarrollar una aplicación web para la gestión global de entidades maestras (Centros, Médicos, Empleados, Especialidades).



Interfaz de Hospitales (React): Desarrollar una aplicación web para la gestión de Consultas Médicas. Debe aplicar autenticación para restringir el acceso y la escritura de datos al centro correspondiente (apoyando la Fragmentación Horizontal).

Integración: Ambos frontends deben consumir la API RESTful de forma segura y eficiente.

Parte 4: Despliegue, Pruebas y Documentación Final

Despliegue y CI/CD: Gestionar el proyecto en Azure DevOps (control de versiones, pipelines CI/CD). Desplegar las VMs y/o contenedores Docker en Azure. Configurar Azure Load Balancer para escalabilidad.

- Pruebas Avanzadas: Realizar pruebas de integración que validen:
 - La correcta replicación de datos globales (Maestro → Esclavos).
 - La efectividad de la fragmentación de datos locales (Consultas).
 - La funcionalidad completa de los endpoints CRUD.

Documentación Técnica Final: Entregar una documentación exhaustiva que incluya:

- Diagramas de Arquitectura (MVC y Flujo de Datos Distribuidos).
- Justificación del modelo de distribución (Centralizado vs. Distribuido, Fragmentación Híbrida).
- Detalles de la configuración de Docker y el esquema de Replicación.
- Documentación de la API (Swagger).

2.5 Listado de equipos, materiales y recursos

Listado de equipos y materiales generales empleados en la guía práctica:

- Laptops
- PostgreSQL
- Visual Studio

TAC (Tecnologías para el Aprendizaje y Conocimiento) empleados en la guía práctica:

- Plataformas educativas
- Simuladores y laboratorios virtuales
- Aplicaciones educativas
- Recursos audiovisuales
- Gamificación
- Inteligencia Artificial



Otros (Especifique): _____

2.6 Actividades desarrolladas

PARTE 1: CONFIGURACIÓN DE LA INFRAESTRUCTURA

MAQUINAS VIRTUALES EN LA NUBE

Se hará uso de Máquinas Virtuales (VMs) en la nube (Azure) para simular el entorno distribuido del sistema hospitalario. Este enfoque es fundamental porque permite replicar con precisión la infraestructura física de los centros médicos y probar las tecnologías de bases de datos distribuidas en un entorno real y aislado.

Para la fase de configuración de la infraestructura, se empleará VirtualBox como hypervisor para crear y gestionar las máquinas virtuales que representarán a los centros médicos.

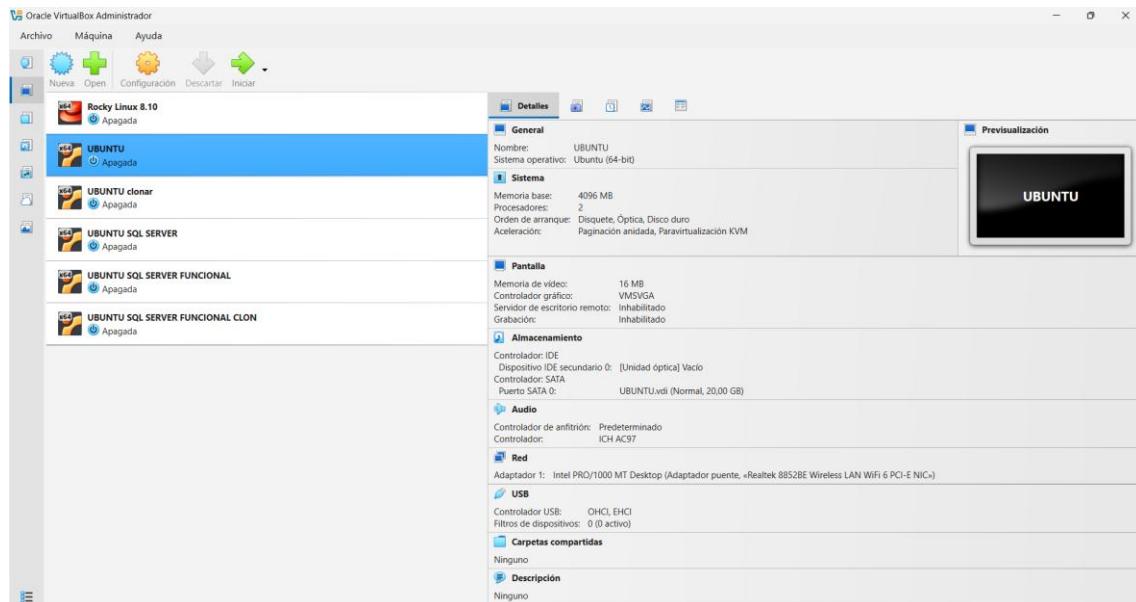


Ilustración 1 Virtual Box

BASES DE DATOS DISTRIBUIDAS Y ASPECTOS AVANZADOS

La selección del SGBD y los modelos de distribución y replicación se basan en la necesidad de optimizar el rendimiento de la transacción local, asegurar la alta disponibilidad de los datos maestros y simplificar la gestión en un entorno distribuido.

1.- SELECCIÓN DEL SGBD

Se utilizará PostgreSQL en todas las Máquinas Virtuales. Se utilizará PostgreSQL en todas las Máquinas Virtuales.

¿Por qué se seleccionó PostgreSQL?

Es una de las bases de datos relacionales más robustas y avanzadas del mercado, con un fuerte cumplimiento ACID y excelentes capacidades para manejo de datos complejos. Es ideal para un sistema hospitalario donde la integridad de los datos (historias clínicas, diagnósticos) es crítica. Además, su arquitectura permite una fácil configuración de



rélicas (mediante Streaming Replication) y soporta herramientas de fragmentación avanzada.

2.- TIPO DE BASE DE DATOS

Como antes fue mencionado en el punto 1 en todas las máquinas virtuales se hará uso de PostgreSQL por lo que será un sistema de bases de datos **HOMOGENEO**.

3.- MODELO DE DISTRIBUCIÓN

Se adoptará un Modelo **Distribuido/Híbrido** que combina una autoridad central para datos maestros con autonomía local para las transacciones.

¿Por qué un Modelo Distribuido?

A diferencia de un modelo puramente centralizado (que crea un punto único de fallo y congestión en Quito), el modelo distribuido mejora la disponibilidad y la escalabilidad. Si la conexión de red con Quito falla, los centros locales (Guayaquil, Cuenca) pueden seguir registrando Consultas Médicas localmente (gracias a la Fragmentación Horizontal) y continuar sus operaciones de lectura usando las rélicas locales de los datos maestros.

¿Por qué un modelo de distribución Híbrido?

Se implementa la fragmentación más avanzada (Híbrida) para optimizar el rendimiento y la seguridad: Fragmentación Horizontal para las tablas de transacciones (Consultas Médicas), dividiendo las filas por la clave Ciudad (centro de origen). Fragmentación Vertical para las tablas sensibles (Empleados, Médicos), separando columnas operativas (ID, Nombre) de columnas sensibles (Salario, Cédula), lo que mejora la seguridad y reduce la latencia de lectura en las rélicas.

ESQUEMA DE DISTRIBUCIÓN HIBRIDA A IMPLEMENTAR



DISEÑO DE FRAGMENTACIÓN MIXTA

TABLAS GLOBALES: Especialidades y AsignacionEspecialidades

EspecialidadID	Nombre	Descripción	MedicodID	AsignacionID
----------------	--------	-------------	-----------	--------------

FRAGMENTACIÓN HORIZONTAL POR CIUDAD

QUITO

o Ciudad='QUITO'

GUAYAQUIL

o Ciudad='GUAYAQUIL'

CUENCA

o Ciudad='CUENCA'

FRAGMENTACIÓN VERTICAL EN CADA CIUDAD

QUITO

V1 Identificación

CentroID (PK)

Nombre

Ciudad

EmpleadodID (PK)

MedicodID (PK)

ClientelID (PK)

ConsultalD (PK)

Nombres

Apellidos

V2 Detalle

CentroID (PK)

Direccion

Telefono

Email

Cargo

Correo

GUAYAQUIL

V1 Identificación

CentroID (PK)

Nombre

Ciudad

EmpleadodID (PK)

MedicodID (PK)

ClientelID (PK)

ConsultalD (PK)

Nombres

Apellidos

V2 Detalle

CentroID (PK)

Direccion

Telefono

Email

Cargo

Correo

CUENCA

V1 Identificación

CentroID (PK)

Nombre

Ciudad

EmpleadodID (PK)

MedicodID (PK)

ClientelID (PK)

ConsultalD (PK)

Nombres

Apellidos

V2 Detalle

CentroID (PK)

Direccion

Telefono

Email

Cargo

Correo

V3 Datos Clínicos

ConsultalD (PK)

MedicodID (FK)

ClientelID (FK)

FechaConsulta

Diagnóstico

Tratamiento

EspecialidadID (FK)

V3 Datos Clínicos

ConsultalD (PK)

MedicodID (FK)

ClientelID (FK)

FechaConsulta

Diagnóstico

Tratamiento

EspecialidadID (FK)

V3 Datos Clínicos

ConsultalD (PK)

MedicodID (FK)

ClientelID (FK)

FechaConsulta

Diagnóstico

Tratamiento

EspecialidadID (FK)

Fragmentación Mixta: Horizontal por Ciudad + Vertical por tipo de datos (Identificación, Detalle, Datos Clínicos)

Ilustración 2 Diagrama de Fragmentación Mixta a Implementar

4.- ESQUEMAS DE REPLICACIÓN

Se elegirá una replicación **PASIVA (MAESTRO-ESCLAVO) Y ASINCRÓNICA** para priorizar el rendimiento y la disponibilidad.



¿Por qué una replicación Pasiva (Maestro – Esclavo)?

Debido a que se designa a Quito como el nodo Maestro para todas las operaciones de escritura en los datos globales (Médicos, Empleados). Los nodos de Guayaquil y Cuenca son Esclavos de Lectura para estos datos. Este esquema simplifica la gestión de la consistencia, ya que todas las escrituras globales pasan por un punto único, evitando conflictos de actualización.

¿Por qué una Replicación Asincrónica?

Ya que la réplica se realiza sin esperar la confirmación del esclavo. Esto es crucial en un sistema hospitalario multi-centro: mejora drásticamente el rendimiento de la aplicación (baja latencia) para las operaciones de escritura en el Maestro. Se acepta un ligero desfase temporal (lag) entre el Maestro y los Esclavos, el cual es tolerable para los datos maestros que no requieren coherencia inmediata (ej., un nuevo médico se registra en Quito y aparece unos segundos después en Guayaquil).

PARTE 2.- IMPLEMENTACIÓN DE LA BASE DE DATOS

Antes de iniciar con la implementación primero debemos tener un esquema de como se va a realizar la base de datos.

ESQUEMA DE LA BASE DE DATOS A IMPLEMENTAR

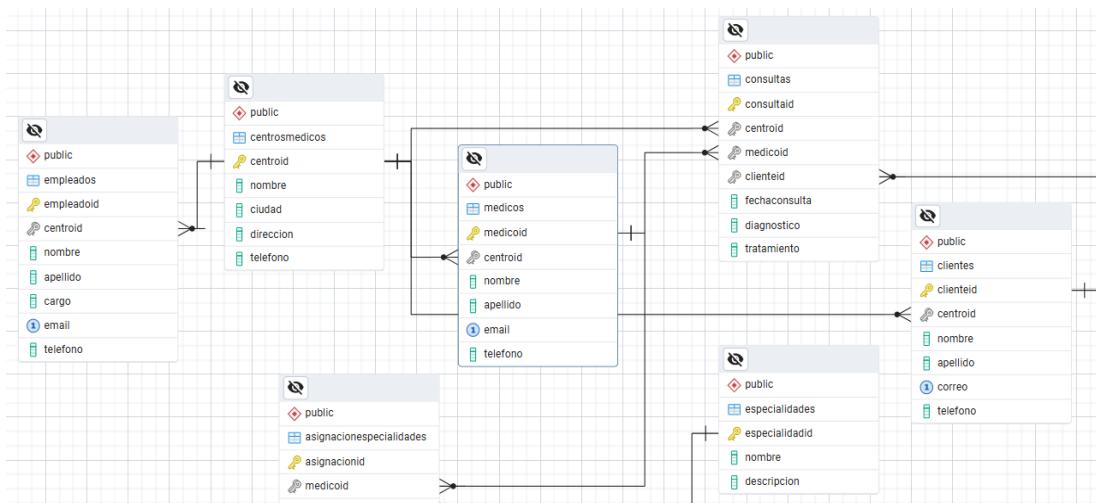


Ilustración 3 Esquema de la base de datos

IMPLEMENTACIÓN DE SERVIDOR, BASE DE DATOS, TABLAS, INSERCIÓN DE DATOS

1.- Creación de un grupo de servidores dentro de PostgreSQL simulando un nodo.

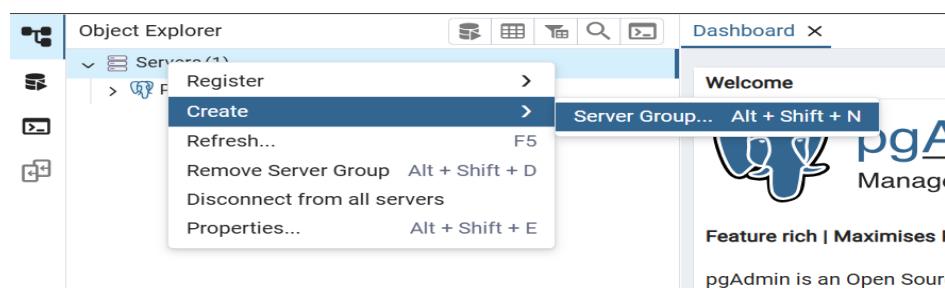


Ilustración 4 Creación de Nodo en PostgreSQL



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



Create - Server Group

General

Name

Ilustración 5 Asignación del Nombre al Nodo creado

Verificación de la correcta creación

Object Explorer

>
>

Ilustración 6 verificación de la correcta creación

2.- Añadir un nuevo servidor dentro del grupo de servidores creado

Quick Links

Ilustración 7 Añadir un nuevo servidor

Seleccionar el grupo de servidores dentro del cual se va a crear el nuevo.

Register - Server

General Connection Parameters SSH Tunnel Advanced Post Connection SQL Tags

Name

Server group

Background

Foreground

Connect now?

Comments

! Either Host name or Service must be specified.

Ilustración 8 Seleccionar el grupo de servidores dentro del cual se va a crear



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



Configuración del host, puerto y contraseña del nuevo servidor

The screenshot shows the 'Register - Server' dialog with the 'Connection' tab selected. The configuration fields are as follows:

- Host name/address: localhost
- Port: 5432
- Maintenance database: postgres
- Username: postgres
- Kerberos authentication?: Off (disabled)
- Password: (redacted)
- In edit mode the password field is enabled only if Save Password is set to true.
- Save password?: On (enabled)
- Role: (empty)
- Service: (empty)

At the bottom are buttons for Close, Reset, and Save.

Ilustración 9 Configuración del host, puerto y contraseña

Instancia (Nodo) Creado correctamente

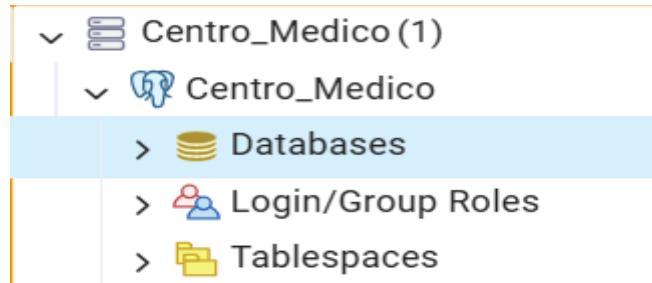


Ilustración 10 Instancia Creada

3.- Creación de la Base de Datos

The screenshot shows the 'Create - Database' dialog with the 'General' tab selected. The configuration fields are as follows:

- Database: sistema_hospitalario
- OID: (empty)
- Owner: postgres
- Comment: (empty)

At the bottom are buttons for Close, Reset, and Save.

Ilustración 11 Creación y Asignación de nombre a la nueva base



4.- Creación de Tablas

Una vez creada correctamente nuestra base de datos dentro de ella crearemos las tablas necesarias.

Creación de la tabla CentrosMedicos

Query Query History

```
1 CREATE TABLE CentrosMedicos (
2     CentroID SERIAL PRIMARY KEY,
3     Nombre VARCHAR(100) NOT NULL,
4     Ciudad VARCHAR(100) NOT NULL,
5     Direccion VARCHAR(150),
6     Telefono VARCHAR(20)
7 );
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 76 msec.

Ilustración 12 Creación de la tabla CentrosMedicos

Creación de la tabla Empleados

Query Query History

```
1 CREATE TABLE Empleados (
2     EmpleadoID SERIAL PRIMARY KEY,
3     CentroID INT NOT NULL,
4     Nombre VARCHAR(100) NOT NULL,
5     Apellido VARCHAR(100) NOT NULL,
6     Cargo VARCHAR(50) NOT NULL,
7     Email VARCHAR(120) UNIQUE NOT NULL,
8     Telefono VARCHAR(20),
9     CONSTRAINT fk_empleados_centro
10    FOREIGN KEY (CentroID) REFERENCES CentrosMedicos (CentroID)
11    ON DELETE CASCADE
12 );
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 71 msec.

Ilustración 13 Creación de la tabla Empleados



UNIVERSIDAD TÉCNICA DE AMBATO
FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL
CARRERA DE Elige un elemento.
CICLO ACADÉMICO: AGOSTO 2025 - ENERO 2026



Creación de la tabla Medicos

```
Query   Query History
1 CREATE TABLE Medicos (
2     MedicoID SERIAL PRIMARY KEY,
3     CentroID INT NOT NULL,
4     Nombre VARCHAR(100) NOT NULL,
5     Apellido VARCHAR(100) NOT NULL,
6     Email VARCHAR(120) UNIQUE NOT NULL,
7     Telefono VARCHAR(20),
8     CONSTRAINT fk_medicos_centro
9         FOREIGN KEY (CentroID) REFERENCES CentrosMedicos (CentroID)
10        ON DELETE CASCADE
11    );
Data Output  Messages  Notifications
CREATE TABLE
Query returned successfully in 100 msec.
```

Ilustración 14 Creación de la tabla Medicos

Creación de la tabla Especialidades

```
Query   Query History
1 CREATE TABLE Medicos (
2     MedicoID SERIAL PRIMARY KEY,
3     CentroID INT NOT NULL,
4     Nombre VARCHAR(100) NOT NULL,
5     Apellido VARCHAR(100) NOT NULL,
6     Email VARCHAR(120) UNIQUE NOT NULL,
7     Telefono VARCHAR(20),
8     CONSTRAINT fk_medicos_centro
9         FOREIGN KEY (CentroID) REFERENCES CentrosMedicos (CentroID)
10        ON DELETE CASCADE
11    );
Data Output  Messages  Notifications
CREATE TABLE
Query returned successfully in 100 msec.
```

Ilustración 15 Creación de la tabla Especialidades

Creación de la tabla AsignacionEspecialidades

```
Query   Query History
1 CREATE TABLE AsignacionEspecialidades (
2     AsignacionID SERIAL PRIMARY KEY,
3     MedicoID INT NOT NULL,
4     EspecialidadID INT NOT NULL,
5     CONSTRAINT fk_asignacion_medico
6         FOREIGN KEY (MedicoID) REFERENCES Medicos (MedicoID)
7         ON DELETE CASCADE,
8     CONSTRAINT fk_asignacion_especialidad
9         FOREIGN KEY (EspecialidadID) REFERENCES Especialidades (EspecialidadID)
10        ON DELETE CASCADE,
11     CONSTRAINT uq_medico_especialidad UNIQUE (MedicoID, EspecialidadID)
12    );
Data Output  Messages  Notifications
CREATE TABLE
Query returned successfully in 62 msec.
```

Ilustración 16 Creación de la tabla AsignacionEspecialidades



Creación de la tabla Clientes

```
Query  Query History
1 CREATE TABLE Clientes (
2     ClienteID SERIAL PRIMARY KEY,
3     CentroID INT NOT NULL,
4     Nombre VARCHAR(100) NOT NULL,
5     Apellido VARCHAR(100) NOT NULL,
6     Correo VARCHAR(120) UNIQUE NOT NULL,
7     Telefono VARCHAR(20),
8     CONSTRAINT fk_clientes_centro
9         FOREIGN KEY (CentroID) REFERENCES CentrosMedicos (CentroID)
10        ON DELETE CASCADE
11 );
Data Output  Messages  Notifications
CREATE TABLE
Query returned successfully in 80 msec.
```

Ilustración 17 Creación de la tabla Clientes

Creación de la tabla Consultas

```
Query  Query History
1 CREATE TABLE Consultas (
2     ConsultaID SERIAL PRIMARY KEY,
3     CentroID INT NOT NULL,
4     MedicoID INT NOT NULL,
5     ClienteID INT NOT NULL,
6     FechaConsulta TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
7     Diagnostico TEXT,
8     Tratamiento TEXT,
9     CONSTRAINT fk_consulta_centro
10        FOREIGN KEY (CentroID) REFERENCES CentrosMedicos (CentroID)
11        ON DELETE CASCADE,
12     CONSTRAINT fk_consulta_medico
13        FOREIGN KEY (MedicoID) REFERENCES Medicos (MedicoID)
14        ON DELETE CASCADE,
15     CONSTRAINT fk_consulta_cliente
16        FOREIGN KEY (ClienteID) REFERENCES Clientes (ClienteID)
17        ON DELETE CASCADE
18 );
Data Output  Messages  Notifications
CREATE TABLE
Query returned successfully in 61 msec.
```

Ilustración 18 Creación de la tabla Consultas

Verificación de la correcta creación de las tablas

▼	Tables (7)
>	asignacionesespecialidades
>	centrosmedicos
>	clientes
>	consultas
>	empleados
>	especialidades
>	medicos

Ilustración 19 Tablas creadas



2.7 Resultados obtenidos

Se ha logrado establecer la estructura fundamental y la arquitectura de distribución avanzada del sistema hospitalario integrado. Los resultados clave incluyen la definición de la infraestructura con la designación de tres Máquinas Virtuales (VMs) (Quito, Guayaquil, Cuenca) para simular los centros médicos, configuradas bajo un entorno de red seguro (VNet/NSG simulado en VirtualBox para desarrollo, y Azure para despliegue). Se ha justificado la elección de PostgreSQL como SGBD homogéneo y se ha seleccionado un Modelo Distribuido/Híbrido que emplea Fragmentación Horizontal para las transacciones (Consultas Médicas) y una potencial Fragmentación Vertical para los datos sensibles (Médicos, Empleados). Finalmente, se ha definido un esquema de Replicación Pasiva y Asincrónica para los datos globales, lo que garantiza la alta disponibilidad y un rendimiento óptimo en la inserción de datos, completando así la fase de diseño de la persistencia y distribución.

2.8 Habilidades blandas empleadas en la práctica

- Liderazgo
- Trabajo en equipo
- Comunicación asertiva
- La empatía
- Pensamiento crítico
- Flexibilidad
- La resolución de conflictos
- Adaptabilidad
- Responsabilidad

2.9 Conclusiones

- Se diseñó correctamente la fragmentación mixta por sede y tipo de datos, incluyendo AlumnoID en ambos fragmentos verticales para garantizar la reconstrucción completa de la información.
- Se implementaron exitosamente los 3 esquemas con sus tablas fragmentadas, constraints y vistas. La vista global permite acceder a todos los datos de forma transparente.
- Las consultas ejecutadas sobre la vista global funcionaron correctamente, confirmando que la reconstrucción mediante JOIN y UNION ALL opera sin pérdida de datos.

2.10 Recomendaciones

- Mantener el atributo AlumnoID como clave común en todos los fragmentos verticales, ya que es fundamental para garantizar la integridad y reconstrucción de los datos.
- Documentar claramente la correspondencia entre fragmentos y vistas para facilitar mantenimiento y futuras ampliaciones del sistema.
- En entornos con múltiples servidores físicos, evaluar la replicación o federación de datos para mantener coherencia entre sedes y mejorar la eficiencia de consultas distribuidas.



- Usar UNION ALL en lugar de UNION para evitar costos innecesarios de eliminación de duplicados, siempre que se garantice que no haya solapamiento de datos entre sedes.

2.11 Anexos

