Evaluación Práctica: Desarrollo de un Sistema de Gestión Hospitalaria en Microservicios

Objetivo: Desarrollar una aplicación distribuida utilizando la arquitectura de microservicios, donde se gestione la información de un sistema hospitalario con múltiples centros médicos, aplicando los principios de bases de datos distribuidas, replicación y servicios web. Los estudiantes deben implementar tanto el backend como el frontend, y garantizar la comunicación entre los microservicios y la integración de las bases de datos.

Parte 1: Configuración de la Infraestructura

1. **Máquinas Virtuales en la Nube**:
    - Configura al menos tres máquinas virtuales (VM), donde cada VM representará un centro médico. La infraestructura debe permitir la comunicación entre ellas, replicando la base de datos desde una máquina central a las máquinas de los centros médicos.
    - La máquina principal será la encargada de gestionar la base de datos central, mientras que las máquinas de los centros médicos tendrán bases de datos locales.
2. **Base de Datos Distribuida**:
    - Configura MariaDB u Otra SQL, en cada máquina para manejar la replicación entre los centros médicos y la base de datos central.
    - Implementa la replicación unidireccional de la base de datos para que los hospitales puedan almacenar sus consultas de manera independiente, pero consulten la base de datos central para otras entidades (empleados, médicos, especialidades).
3. **Acceso Remoto a la Base de Datos**:
    - Configura el acceso remoto a las bases de datos para permitir la gestión desde la nube y las consultas distribuidas.

Parte 2: Desarrollo de la Aplicación Backend

1. **API de Administración**:
    - Desarrolla una API utilizando Node.js y Express que permita la gestión de los centros médicos, médicos, especialidades y empleados.
    - Implementa las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) para cada entidad (Centro Médico, Médico, Empleado, Especialidad).
    - Asegúrate de que la API sea capaz de interactuar con la base de datos distribuida y realizar consultas a la base de datos central y a las locales de cada hospital.
2. **API de Consultas Médicas**:
    - Desarrolla un servicio web que permita a los hospitales crear, consultar, actualizar y eliminar registros de consultas médicas, asegurando que estos se almacenen de forma independiente en cada hospital.

- Los hospitales deben poder generar reportes de consultas por médico, y la administración debe poder consultar estos datos desde una aplicación web central.

3. **Documentación del Backend**:
   - Documenta la API utilizando herramientas como Swagger para proporcionar una referencia clara de los endpoints disponibles y cómo interactuar con la API.

Parte 3: Desarrollo del Frontend

1. **Interfaz de Usuario (UI) de Administración**:
   - Desarrolla una interfaz web para la administración que permita gestionar los centros médicos, empleados, médicos y especialidades.
   - Implementa funcionalidades como la creación de nuevos registros, la edición y eliminación de entidades, y la visualización de estadísticas como nóminas y reportes.
2. **Interfaz de Usuario (UI) para los Hospitales**:
   - Desarrolla una interfaz web para los hospitales donde los médicos puedan gestionar consultas, ver información de especialidades y realizar tareas relacionadas con la atención al paciente.
   - Implementa la autenticación de usuario, permitiendo que los médicos accedan solo a la información relevante de su centro médico.
3. **Despliegue de Aplicación**:
   - Despliega ambas aplicaciones (frontend y backend) en una plataforma en la nube como Heroku o AWS, para garantizar su disponibilidad y acceso desde diferentes ubicaciones.

Parte 4: Implementación de Microservicios

1. **Arquitectura de Microservicios**:
   - Implementa una arquitectura de microservicios para que cada componente del sistema (administración, consulta médica, gestión de usuarios, etc.) esté desacoplado y pueda escalar independientemente.
   - Asegura la comunicación entre los microservicios utilizando RESTful APIs.
2. **Comunicación entre Servicios**:
   - Implementa mecanismos de comunicación entre los microservicios, utilizando tecnologías como gRPC, RabbitMQ, o Kafka para la mensajería asíncrona si es necesario.
   - Los servicios deben ser capaces de comunicarse entre sí para compartir datos de manera eficiente y sin dependencia directa.

Parte 5: Evaluación y Reporte Final

1. **Pruebas de Integración**:
    - Realiza pruebas de integración para verificar que los microservicios interactúan correctamente y que los datos se replican y sincronizan adecuadamente entre las bases de datos distribuidas.
    - Verifica que la información de los hospitales se maneja de manera independiente, pero con la capacidad de generar reportes centralizados.
2. **Documentación Técnica**:
    - Entrega una documentación técnica detallada sobre cómo se configuró el sistema, las decisiones arquitectónicas tomadas, y cómo se integraron los diferentes microservicios.
    - Incluye diagramas de la arquitectura, las configuraciones de las bases de datos, y los procedimientos para la puesta en producción.

Criterios de Evaluación:

- **Configuración de la Infraestructura**: Correcta implementación de bases de datos distribuidas y la replicación entre ellas.
- **Desarrollo Backend**: Implementación de APIs RESTful eficientes y funcionales con operaciones CRUD completas.
- **Desarrollo Frontend**: Interfaces intuitivas y completas para la administración y gestión hospitalaria.
- **Arquitectura de Microservicios**: Correcto diseño y comunicación entre microservicios.
- **Despliegue y Mantenimiento**: La capacidad de mantener y desplegar las aplicaciones en la nube de forma exitosa.
- **Pruebas y Documentación**: Realización de pruebas de integración y documentación detallada.

ARQUITECTURA PROPUESTA:

La arquitectura propuesta para este proyecto se basa en un sistema distribuido de microservicios. Estos microservicios estarán conectados a bases de datos distribuidas y se desplegarán en Azure, con servicios de contenedores y máquinas virtuales. Aquí está el diagrama de alto nivel de la arquitectura:

1. **Máquinas Virtuales (VMs)**:
    - **Centro Médico Quito (Admin)**: Base de datos principal y servidor de administración.
    - **Centro Médico Guayaquil**: Base de datos de consultas de Guayaquil.
    - **Centro Médico Cuenca**: Base de datos de consultas de Cuenca.

- **Base de Datos Distribuida**: MariaDB configurado para replicación entre las diferentes bases de datos.

2. **Base de Datos**:
   - **Base de Datos Central (Quito)**: Almacena los registros globales (empleados, médicos, especialidades, centros médicos) con MariaDB.
   - **Bases de Datos Locales (Guayaquil y Cuenca)**: Bases locales de MariaDB donde se almacenan las consultas médicas locales. Estas bases están replicadas desde Quito.
3. **Backend y Microservicios**:
   - **Microservicios de Administración**: Desarrollados en Node.js y Express, gestionan los registros globales (empleados, médicos, especialidades, centros médicos).
   - **Microservicios de Consultas Médicas**: Manejan las consultas médicas locales en cada hospital.
   - **API Gateway**: Gestiona la comunicación entre el frontend y los microservicios.
4. **Frontend**:
   - **Interfaz Administrativa**: Desarrollada en Vue.js, esta interfaz permite gestionar las entidades globales.
   - **Interfaz de Hospital**: Desarrollada en React, esta interfaz permite a los hospitales gestionar consultas médicas.
5. **Azure Services**:
   - **Azure VMs**: Se usan para hospedar los microservicios y las bases de datos.
   - **Azure Load Balancer**: Para gestionar el tráfico entre diferentes instancias de microservicios.
   - **Azure App Services**: Para desplegar las aplicaciones web de administración y de los hospitales.
6. **Red y Seguridad**:
   - **Red Virtual (VNet)**: Para garantizar que todas las máquinas virtuales estén conectadas de manera segura.
   - **Grupos de Seguridad de Red (NSG)**: Para controlar el acceso entre las máquinas virtuales y la base de datos.

Pasos para Construir la Arquitectura en Azure

1. Crear Máquinas Virtuales en Azure

1. **Accede a tu cuenta de Azure** y ve al portal de **Azure**.

2. Crea tres **máquinas virtuales (VM)** para los centros médicos:
    - **Centro Médico Quito**: Esta será la VM principal con la base de datos central.
    - **Centro Médico Guayaquil**: Esta será la VM para la base de datos de Guayaquil.
    - **Centro Médico Cuenca**: Esta será la VM para la base de datos de Cuenca.

Puedes utilizar **Ubuntu 20.04 LTS** como sistema operativo para las máquinas virtuales.

2. Configuración de MariaDB en las Máquinas Virtuales (puede ser PostgreSQL, Mysql…)

1. **Instalar MariaDB** en cada máquina virtual:

```
sudo apt update
sudo apt install mariadb-server
sudo mysql_secure_installation
```

2. **Configurar la base de datos central (Quito)**:
    - Crea la base de datos centro_medico y configura la replicación.
    - Edita el archivo de configuración de MariaDB para habilitar la replicación.
3. **Configurar la replicación**:
    - En la máquina **Quito**, configura la replicación como *maestro*.
    - En las máquinas **Guayaquil** y **Cuenca**, configura la replicación como *esclavos*.
4. **Verifica la replicación** utilizando los comandos SHOW MASTER STATUS y SHOW SLAVE STATUS.

3. Crear un API Gateway y Microservicios

1. **Crea una nueva aplicación de microservicios en Azure**:
    - **Microservicios de administración**: Desarrollados con Node.js y Express, que gestionan centros médicos, médicos, especialidades y empleados.
    - **Microservicios de consultas médicas**: Desarrollados con Java, que gestionan las consultas médicas locales.
2. **Desplegar los microservicios en Azure App Services**:
    - Crea un servicio de **Azure App Service** para el backend de administración y otro para los hospitales.
    - Conecta los microservicios a las bases de datos correspondientes.
3. **API Gateway**:
    - Configura un **API Gateway** para gestionar las solicitudes entre el frontend y los microservicios.

4. Desarrollar y Desplegar el Frontend

1. **Desarrollar el frontend con Vue.js para la administración**:

- Crea una aplicación web que permita gestionar los registros de los centros médicos, médicos, especialidades y empleados.

2. **Desarrollar el frontend con React para los hospitales**:
   - Desarrolla una interfaz para gestionar consultas médicas, donde los hospitales puedan registrar, actualizar y consultar las citas.
3. **Desplegar las aplicaciones frontend**:
   - Utiliza **Azure App Service** o **Azure Static Web Apps** para desplegar ambas aplicaciones web en la nube.

5. Configuración de la Red y Seguridad

1. **Crear una Red Virtual (VNet)**:
   - Asegúrate de que todas las máquinas virtuales estén dentro de la misma red virtual en Azure para garantizar la comunicación segura.
2. **Configurar los Grupos de Seguridad de Red (NSG)**:
   - Asegura las máquinas virtuales y las aplicaciones frontend/backend mediante NSG para controlar el acceso a las bases de datos y a los servicios.
3. **Configurar Azure Load Balancer**:
   - Si es necesario, utiliza **Azure Load Balancer** para distribuir el tráfico entre múltiples instancias de tus microservicios.

6. Despliegue Final y Pruebas

1. **Prueba la replicación de la base de datos**: Verifica que los datos se replican correctamente entre Quito, Guayaquil y Cuenca.
2. **Prueba los microservicios**: Asegúrate de que las operaciones CRUD en los microservicios funcionen correctamente para las entidades hospitalarias.
3. **Prueba la interfaz web**: Asegúrate de que los usuarios de administración y los hospitales puedan acceder y gestionar las consultas médicas correctamente.

CONSIDERACIONES:

- CONSTRUIR EN SOBRE AZURE DEVOPS.
- COMPARTIR TODA LA DOCUMENTACION EN AZURE DEVOPS
- FECHA DE ENTREGA 18 DE ABRIL DEL 2025