PARCIAL CICLO VIDA DEL DESARROLLO DE SOFTWARE TERCER TERCIO (CVDS)

Contexto General

La clínica ECI Salud Vital necesita una aplicación web simple para gestionar citas médicas. Como desarrollador, deberás implementar un sistema que permita a los usuarios ver especialidades médicas, programar citas y consultar su historial de citas. El sistema debe ser funcional, con backend desplegado en Azure, frontend local, y persistencia en MongoDB.

Requerimientos

1. **Página de Inicio**:
    - Mostrará 4 especialidades médicas: (medicina general, psicología, ortopedia, odontologia)
        - Nombre e imagen en un esquema 2x2.
    - Al seleccionar una especialidad, mostrará:
        - Imagen ampliada.
        - Descripción.
        - Doctor que la atiende.
        - Ubicación.
        - Botón para programar una cita.
2. **Programar Cita**:
    - Formulario con:
        - Nombre completo.
        - Cedula.
        - Correo electrónico.
    - Resumen de la cita:
        - Fecha de la cita (DD-MM-YYYY).
        - Especialidad seleccionada.
        - Doctor que la atiende.
        - Ubicación.
    - Botón "Confirmar Cita":
        - Valida que los campos no estén vacíos.
        - Si falla, muestra un error y marca la cita como "Rechazada". (Si la fecha en que se desea la cita no es válida)
        - Si es exitosa, marca la cita como "Confirmada" y la almacena.
3. **Historial de Citas**:
    - Lista de tarjetas de las citas del usuario (identificado por correo).
    - Mostrar: ID de cita, especialidad, fecha, estado (Confirmada en verde)
    - Filtrar por estado (Confirmada/Cancelada).
4. **Criterios de Aceptación**:
    - Si se cancela una cita (en el historial) el estado cambiará a Cancelada y se visualizará en color Rojo.
    - Todos los cambios se reflejan en MongoDB.

- Validar que los datos del formulario sean correctos; si no, mostrar un mensaje de error en la cita.
- Las citas médicas no pueden programarse con fechas anteriores al día del parcial.

Puntos a Evaluar

1. **Diseño de Arquitectura**:
   - Crear un **diagrama de componentes, clases y datos de la solución** e incluirlo en el README junto con un ejemplo de las colecciones en formato JSON.
2. **Implementación**:
   - **Backend**:
     - Endpoints:
       - Especialidades
       - Programar cita con validación.
       - Filtrar cita por estado.
       - Cancelar cita
     - Implementar Swagger para documentar endpoints.
   - **Frontend**:
     - Interfaz para mostrar especialidades (2x2), detalle de especialidad, formulario de cita, historial de citas.
     - Conectar al backend desplegado en Azure.
3. **Pruebas**:
   - Pruebas unitarias para los servicios (CRUD, programar cita, cancelar cita) con cobertura mínima del **60%**.
4. **Despliegue**:
   - Desplegar el backend en Azure (Azure App Service).
   - Frontend conectado al backend que apunta al Azure.
5. **Documentación**:
   - **README Backend**: Nombre del estudiante, grupo, diagramas de arquitectura, clases y datos, instrucciones para ejecutar, capturas de pantalla de la cobertura de pruebas y capturas de pantalla en postman, link a Swagger, lista de endpoints.
   - **README Frontend**: Nombre, grupo, tecnologías, instrucciones, Capturas de pantalla explicadas.

OBSERVACIONES:

1. **Si no tiene REAMDE el parcial no será revisado.**
2. **Quien sea visto utilizando Teams, Slack o cualquier otro medio de comunicación para hacer copia, su parcial se anulará inmediatamente.**
3. **No está permitido el uso de celulares dentro de la sesión.**
4. **Los commit deben reflejar el trabajo y esfuerzo realizado, por tanto si se considera sospechoso podrá ser causal de anulación del parcial.**