Enunciado de la Prueba de desempeño

C#

Caso de uso:
El **Hospital San Vicente** actualmente gestiona sus citas médicas en agendas físicas y hojas de cálculo. Este método ha generado múltiples problemas:

- Duplicidad de citas para el mismo médico o paciente.
- Dificultad para encontrar la información de un paciente.
- Falta de control sobre médicos y sus especialidades.
- Pérdida de información cuando las agendas se dañan o extravían.

La gerencia del hospital decidió desarrollar un **sistema interno** en **c#** que permita organizar de forma eficiente las **citas médicas, pacientes y médicos**.

Objetivo:
Deberás construir un sistema e implementar una aplicación en C# utilizando Aplicaciones de consola o aplicaciones web, EF Coare, Linq, List<>, Dictionary<TKey, TValue>, MySQL u PostgreSQL que permita al hospital digitalizar y optimizar la gestión de citas médicas, garantizando la integridad, consistencia y accesibilidad de la información.

El sistema deberá:

- **Centralizar la información** de pacientes, médicos y citas, eliminando la dependencia de registros manuales en papel.
- **Facilitar la gestión de pacientes y médicos**, asegurando que los datos estén organizados y disponibles en todo momento.
- **Automatizar la programación de citas médicas**, evitando duplicidades y conflictos de horarios entre pacientes y médicos.
- **Aplicar principios de Programación Orientada a Objetos (POO)**.
- **Incorporar validaciones y manejo de errores**, garantizando que los datos registrados cumplan con las reglas de negocio y que el sistema sea robusto frente a entradas inválidas.

Funcionalidades principales:
Para alcanzar un resultado óptimo en esta prueba, deberás cumplir cada uno de los siguientes requisitos y funcionalidades:

Requisitos:
1. Gestión de Pacientes

- Registrar nuevos pacientes con sus datos personales (nombre, documento, edad, teléfono, correo).

- Editar la información de los pacientes.
- Validar que no existan pacientes duplicados mediante el documento de identidad.
- Listar todos los pacientes registrados en el sistema.

2. Gestión de Médicos

- Registrar médicos con sus datos básicos (nombre, documento, especialidad, teléfono, correo).
- Editar la información de los médicos
- Validar que no existan médicos duplicados (documento único).
- Listar todos los médicos registrados, con opción de filtrar por especialidad.

3. Gestión de Citas Médicas

- **Agendar citas médicas** asignando un paciente, un médico, fecha y hora.
- Validar que:
    - Un médico no tenga más de una cita en el mismo horario.
    - Un paciente no tenga más de una cita en el mismo horario.
- **Cancelar citas** cambiando su estado a *Cancelada*.
- **Marcar citas como atendidas** cambiando su estado a *Atendida*.
- Listar citas médicas por paciente.
- Listar citas médicas por médico.
- Al momento de agendar una cita médica al paciente le llegará un correo de confirmación con la información de su cita.
- Historial de envío de correos electrónicos con su estado “enviado” “no enviado”

4. Persistencia de Datos

- Uso **Listas**, **Dictionary**, **LINQ**, **EFC** para la gestión de la información.

5. Manejo de Errores y Validaciones

- Captura de excepciones mediante bloques try-catch.
- Mensajes de error claros y amigables al usuario.
- Validaciones que garanticen la integridad de los datos según las reglas de negocio.

Criterios de aceptación:
El sistema se considerará **aprobado** si cumple con los siguientes criterios:

1. Gestión de Pacientes

- Es posible registrar un paciente nuevo proporcionando todos los datos obligatorios.
- Es posible editar la información de un paciente.
- El sistema válido que el documento de identidad sea único.
- Se puede visualizar un listado completo de pacientes registrados.

2. Gestión de Médicos

- Es posible registrar un médico con sus datos básicos.
- Es posible editar la información de un médico.
- El sistema, válida que el documento sea único y que no existan médicos con la misma combinación de nombre y especialidad.
- Se puede listar a todos los médicos, con opción de filtrar por especialidad.

3. Gestión de Citas Médicas

- Es posible **agendar una cita** asociando un paciente y un médico en una fecha y hora determinada.
- El sistema impide agendar citas en conflicto:
    - Un médico no puede tener dos citas en el mismo horario.
    - Un paciente no puede tener dos citas en el mismo horario.
- Es posible **cancelar una cita**, y su estado cambia a *Cancelada*.
- Es posible **marcar una cita como atendida**, y su estado cambia a *Atendida*.
- Se puede listar citas médicas por paciente y por médico.

4. Manejo de errores y validaciones

- El sistema muestra mensajes de error claros cuando se ingresan datos inválidos.
- Las excepciones son manejadas mediante bloques try-catch.
- Las reglas de negocio (documentos únicos, no duplicidad de citas, validaciones de especialidad) son respetadas en todos los flujos.

5. Documentación

- El proyecto incluye un archivo **README.md** con:
- Descripción general del sistema.
- Requisitos previos para ejecutar el proyecto (.NET SDK, Bases de datos, etc.).

- Pasos detallados para clonar, configurar y ejecutar la aplicación.
- Capturas de pantalla o ejemplos de vistas.

Entregables:

1. Enlace al repositorio en **GitHub** (público).
2. Proyecto comprimido zip
3. Diagrama de clases.
4. Diagrama de casos de uso.
5. El repositorio debe contener un archivo **README** con instrucciones detalladas sobre el proyecto, además de la información del Coder (**Nombre, Clan, correo, documento de identidad**). Este archivo debe detallar paso a paso cómo levantar y usar la solución, garantizando que el Team Leader no tenga que realizar ingeniería inversa o adivinar cómo el proyecto se corre.