Taller de Programación - TurnosApp

Estudiantes:
- Jesus David Avila Arrieta
- Miguel Ángel Ocampo Loaiza

Profesor: Daniel Andrey Villamizar Araque
Curso: Programación de Software

Introducción

Este taller consiste en desarrollar una aplicación de consola en C# llamada TurnosApp, que permite gestionar turnos de enfermeras. El proyecto aplica principios de Programación Orientada a Objetos (POO), como abstracción, herencia, polimorfismo y encapsulamiento. Además, los datos se guardan en un archivo JSON para asegurar persistencia.

1. Caso de Negocio

En una clínica pequeña hay problemas para organizar los turnos de las enfermeras. Muchas veces se cruzan horarios, se olvidan reglas de descanso y no hay control de cuántos turnos hace cada persona. La aplicación permitirá registrar enfermeras, asignar turnos de día o de noche y validar reglas de descanso. Usuarios: Coordinador de enfermeras. Valor: Mejor organización, menos errores y más transparencia.

2. Historias de Usuario

```
HU1: Como coordinador quiero registrar enfermeras para poder asignarles turnos.
HU2: Como coordinador quiero asignar un turno de día (7–19) o noche (19–7) a una enfermera.
HU3: Como coordinador quiero que no se crucen turnos de una misma enfermera.
HU4: Como coordinador quiero que tras un turno nocturno haya un día de descanso.
HU5: Como coordinador quiero listar las enfermeras y ver cuántos turnos tienen.
HU6: Como coordinador quiero que el sistema guarde la información en archivo JSON para no perder dat
```

3. Requerimientos

```
De negocio:
RN1: Los turnos son de 12 horas exactas.
RN2: Tras turno de noche, hay un día de descanso.
RN3: Ninguna enfermera puede tener turnos cruzados.
RN4: Se debe llevar registro de turnos por enfermera.
RN5: La información debe persistir entre ejecuciones.

Funcionales:
RF1: CRUD de enfermeras.
RF2: Crear turnos día/noche.
```

```
RF3: Validar reglas de descanso.
RF4: Validar choques de turnos.
RF5: Listar enfermeras con sus turnos.
RF6: Guardar y cargar datos en JSON.
RF7: Mostrar menú interactivo en consola.
RF8: Permitir búsqueda por enfermera.

No funcionales:
RNF1: Responder en menos de 1s con 100 registros.
RNF2: Manejo de errores con mensajes claros.
RNF3: Log básico en consola.
RNF4: Código legible con buenas prácticas.
RNF5: Testabilidad con pruebas unitarias.
RNF6: Portabilidad en Windows/Linux con .NET 8.
```

4. Modelo Conceptual

Enfermera: tiene Id, Nombre y una lista de turnos. Turno (abstracto): tiene fecha de inicio y fin. TurnoDia: 7am–7pm. TurnoNoche: 7pm–7am (día siguiente). Relación: Una enfermera puede tener muchos turnos.

5. Diseño POO

Encapsulamiento: La clase Enfermera protege su lista de turnos. Herencia: TurnoDia y TurnoNoche heredan de Turno. Polimorfismo: IAsignadorTurnos permite diferentes estrategias de asignación. Abstracción: Turno es clase abstracta, IAsignadorTurnos es interfaz.

6. Plan de Pruebas

```
Prueba 1: Crear enfermera → debe guardarse con Id único.
Prueba 2: Asignar turno de día → se agrega a la lista de turnos.
Prueba 3: Intentar dos turnos el mismo día → debe dar error.
Prueba 4: Intentar turno después de turno de noche → debe impedir asignación.
Prueba 5: Listar enfermeras → debe mostrar número de turnos.
Prueba 6: Guardar y cargar datos → los turnos deben persistir en JSON.
```

Conclusiones

La aplicación TurnosApp permitió aplicar los principios de POO en C# de manera práctica. Se diseñó un sistema básico pero funcional, con reglas de negocio claras, validación de turnos, persistencia en JSON y pruebas unitarias que verifican su correcto funcionamiento.