Descripción: Deberás desarrollar una aplicación web en TypeScript para gestionar el tradicional sorteo de la cesta de navidad mediante un tablero de 100 números (del 00 al 99). El sistema permitirá a los participantes reservar números y determinará automáticamente el ganador basándose en los dos últimos dígitos del sorteo de la Lotería Nacional. La aplicación debe simular un tablero físico tradicional donde los participantes escriben su nombre en las casillas que desean jugar, con la particularidad de que cada persona puede participar en varios números diferentes, pero cada número sólo puede estar ocupado por un único participante. En muchas empresas, asociaciones y centros educativos se organiza este sorteo navideño donde: ■ Se crea un tablero con 100 casillas numeradas del 00 al 99 ■ Los participantes eligen y "compran" uno o varios números ■ El día del sorteo de la Lotería Nacional, se toman los dos últimos dígitos del primer premio ■ El participante que tenga ese número gana la cesta de navidad Requisitos funcionales: RF1: Gestión de Participantes ■ Permitir registrar participantes con su información básica (nombre, email, teléfono) ■ Validar que no existan participantes duplicados ■ Poder consultar la lista completa de participantes registrados RF2: Gestión del Tablero ■ Crear y mostrar un tablero con los 100 números (00-99) ■ Visualizar claramente qué números están disponibles y cuáles están ocupados ■ Mostrar el nombre del participante que ha reservado cada número ocupado 27 DWECL. UD 2: Estructura del lenguaje JS. Introducción a TS ■ Permitir reservar un número libre para un participante registrado ■ Validar que un número solo pueda ser asignado a un participante a la vez ■ Permitir que un mismo participante reserve múltiples números diferentes ■ Implementar la funcionalidad para liberar un número (cancelar reserva) RF3: Sorteo y Determinación del Ganador ■ Permitir introducir los dos últimos dígitos del sorteo de la Lotería Nacional ■ Determinar automáticamente quién es el ganador según el número premiado ■ Gestionar el caso en que el número premiado no esté ocupado (desierto) ■ Mostrar información del ganador y sus datos de contacto RF4: Consultas e Informes ■ Mostrar estadísticas básicas: números ocupados/libres, número de participantes, etc. ■ Listar todos los números que ha reservado un participante específico ■ Generar un resumen del estado actual del sorteo Requisitos Técnicos RT1: Uso de TypeScript ■ Todo el código debe estar escrito en TypeScript ■ Aprovechar el sistema de tipos para evitar errores en tiempo de ejecución ■ NO usar any salvo casos muy justificados RT2: Modelado mediante Clases ■ Crear al menos una clase Participante para representar a cada persona ■ Crear una clase Tablero o Sorteo que gestione la lógica del juego ■ Implementar métodos adecuados con responsabilidades bien definidas RT3: Interfaces y Tipos Personalizados ■ Definir interfaces para estructurar los datos principales 27 DWECL. UD 2: Estructura del lenguaje JS. Introducción a TS ■ Crear tipos personalizados (type aliases) cuando sea apropiado ■ Usar tipos literales para valores que solo pueden tener opciones específicas RT4: Características Avanzadas de TypeScript ■ Implementar argumentos opcionales en funciones/métodos donde tenga sentido ■ Usar parámetros por defecto cuando corresponda ■ Aplicar union types, intersection types o type guards según necesidad ■ Considerar el uso de genéricos si aporta flexibilidad RT5: Manejo de Errores ■ Implementar validaciones adecuadas ■ Lanzar errores personalizados cuando se produzcan operaciones inválidas ■ Documentar mediante JSDoc los posibles errores que puede lanzar cada método RT6: Testing con Jest (OBLIGATORIO) El proyecto debe incluir una suite completa de tests unitarios utilizando Jest. Este apartado es obligatorio y forma parte de los entregables principales. Configuración de Jest ■ Configurar Jest para trabajar con TypeScript (usar ts-jest) Cobertura Mínima Requerida Debes crear tests para todas las clases principales, cubriendo: ■ Métodos públicos: Todos los métodos públicos deben tener al menos un test ■ Casos de éxito: Verificar que las operaciones funcionan correctamente ■ Casos de error:

Verificar que se lanzan las excepciones esperadas ■ Validaciones: Comprobar que las validaciones funcionan correctamente ■ Casos límite: Probar situaciones extremas (tablero lleno, tablero vacío, etc.) 27 DWECL. UD 2: Estructura del lenguaje JS. Introducción a TS Tests Obligatorios por Funcionalidad Tests para Gestión de Participantes: ■ Crear un participante con datos válidos ■ Intentar crear un participante con datos inválidos (email mal formado, campos vacíos) ■ Verificar que no se pueden registrar participantes duplicados ■ Consultar la lista de participantes Tests para Gestión del Tablero: ■ Inicializar el tablero correctamente con 100 números ■ Reservar un número libre exitosamente ■ Intentar reservar un número ya ocupado (debe fallar) ■ Permitir que un participante reserve múltiples números ■ Liberar un número correctamente ■ Intentar liberar un número que no está ocupado ■ Verificar el estado de números (libre/ocupado) ■ Consultar qué números tiene asignados un participante Tests para Sorteo: ■ Realizar un sorteo con número ganador ocupado ■ Realizar un sorteo con número ganador libre (desierto) ■ Validar que solo se aceptan números entre 00 y 99 ■ Verificar que se devuelve correctamente la información del ganador Tests para Estadísticas: ■ Calcular correctamente el número de casillas ocupadas/libres ■ Contar correctamente el número de participantes únicos ■ Calcular el porcentaje de ocupación Buenas Prácticas en Testing 27 DWECL. UD 2: Estructura del lenguaje JS. Introducción a TS ■ Usar describe para agrupar tests relacionados ■ Nombres descriptivos en los tests (patrón "should..." o "debería...") ■ Aplicar el patrón AAA (Arrange-Act-Assert) en cada test ■ Usar beforeEach y afterEach cuando sea necesario para limpiar estado ■ Utilizar matchers apropiados de Jest (toBe, toEqual, toThrow, toContain, etc.) ■ Crear tests independientes que no dependan del orden de ejecución ■ Utilizar mocks o spies cuando sea necesario (por ejemplo, para simular entrada de usuario) Entregables 1. Código fuente TypeScript organizado en módulos/archivos lógicos 2. Archivo tsconfig.json configurado apropiadamente 3. Documentación técnica que incluya: ■ Diagrama de clases (puede ser simple, hecho con texto ASCII o herramienta) ■ Descripción de las interfaces y tipos principales ■ Explicación de las decisiones de diseño tomadas 4. Ejemplos de uso que demuestren todas las funcionalidades 5. README.md con instrucciones de compilación y ejecución Consejos ■ Empieza por modelar bien las entidades principales antes de escribir código ■ Define primero las interfaces y tipos que necesitarás ■ Implementa funcionalidad básica antes de añadir mejoras ■ Prueba cada método según lo desarrollas ■ Piensa en casos límite: ¿qué pasa si no hay participantes? ¿Y si el número ganador está libre?