

Primer Trabajo para entregar

Seminario de Lenguajes opción .NET
1º Semestre - 2025

Importante

Queremos recordarles la importancia de este primer trabajo del curso. Aunque no recibirá una calificación numérica, será objeto de corrección y dará lugar a una devolución que servirá como guía para continuar hacia el próximo trabajo.

La entrega de este primer trabajo es de carácter obligatorio y requisito excluyente para la aprobación del curso. Se considerará que los alumnos que no realicen esta entrega han decidido abandonar la materia.

La entrega debe ser significativa. No se admitirán entregas en blanco ni aquellas que muestren un esfuerzo mínimo, en lugar de un sincero compromiso y dedicación en la realización del trabajo.

Entrega

Fecha límite: **19/05/2025 hasta las 13:00 hs.**

El trabajo puede realizarse en grupo de hasta 4 integrantes.

La entrega se realizará por medio de un formulario de Google que se publicará más adelante.

Se deberá entregar:

- El código de la solución completa
- Un documento explicativo (puede ser **.pdf** o **.md**) donde se detalle cómo probar la funcionalidad desarrollada desde Program.cs. Se debe explicitar código de ejemplo junto con la salida por consola producida.
- Subir la solución completa y el documento explicativo, todo comprimido en un único archivo (preferentemente **.zip**). El nombre del archivo zip debe contener los apellidos de los autores del trabajo.

Sistema de Gestión del Centro Deportivo Universitario

Objetivo

Se requiere desarrollar un sistema para gestionar las actividades y el uso de instalaciones deportivas dentro de un centro universitario. El sistema permitirá registrar personas (estudiantes o docentes), actividades deportivas, reservas de instalaciones y control de asistencia.

Cada entidad debe tener un identificador único. El sistema debe permitir realizar operaciones básicas como altas, bajas, modificaciones y listados.

Entidades a implementar

- **Persona:** Id, número de carnet, nombre, apellido, dirección, facultad, teléfono, correo electrónico.
- **Estudiante:** número de alumno y carrera.
- **Docente:** matrícula y año de ingreso como docente.
- **ActividadDeportiva:** Id, nombre (fútbol, vóley, natación, etc.), días disponibles, cupo máximo.
- **Reserva:** Id, persona (por Id), actividad deportiva (por Id), fecha de reserva, estado de asistencia (pendiente, asistió, ausente, cancelada).
- **Usuario:** email, contraseña, nombre, permisos

Implementar el método `ToString()` donde sea conveniente.

Definir las interfaces necesarias para trabajar con los repositorios y utilizar inversión de dependencia.

Reglas de negocio

- Una actividad no puede superar su cupo máximo de inscripciones.
- Un participante no puede inscribirse dos veces en la misma actividad.
- No puede modificarse ni cancelarse una actividad ya realizada.
- Los tipos de actividad no pueden eliminarse si hay actividades que lo referencien.
- Las fechas de las actividades deben ser válidas (no anteriores al día actual).

Validaciones

Actividad:

- Nombre y descripción obligatorios.
- Fecha debe ser igual o posterior a la fecha actual.
- El **CupoMaximo** debe ser mayor que cero.
- Debe tener un **TipoActividad** y un **Responsable** válidos.

Responsable:

- Nombre y correo obligatorios.
- DNI no puede repetirse.

Participante:

- Nombre y correo obligatorios.
- DNI no puede repetirse.

Inscripción:

- Fecha válida.
- No permitir duplicados (un mismo participante en la misma actividad).

Excepciones

- **ValidacionException**: Se lanza si algún dato obligatorio está ausente o es incorrecto.
- **CupoExcedidoException**: Se lanza al intentar inscribir más participantes que el cupo disponible.
- **FechaInvalidaException**: Se lanza cuando la fecha de la actividad no es válida.

Implementación

La solución se llamará **CentroEventos** y se dividirá en 3 proyectos:

CentroEventos.Aplicacion

Este proyecto contendrá:

- Las clases de las entidades del sistema.
- Los validadores para cada entidad.
- Interfaces de repositorio. Para acceder a los repositorios usaremos inversión de dependencia, por lo tanto, se deben definir en este proyecto las interfaces necesarias para trabajar con ellos.
- Las excepciones mencionadas
- Casos de uso.

Casos de uso obligatorios

- Altas, bajas, modificaciones y listados de:
 - Personas (Estudiantes/Docentes)
 - Actividades
 - Reservas

Las operaciones de eliminación, se llevarán a cabo utilizando el Id de la entidad que se desea eliminar y que se pasa como parámetro.

Clase `ReservarActividadUseCase` con método: `void Ejecutar(Reserva reserva)`

- Registra una reserva. Validar que haya cupo disponible. En caso contrario, lanzar excepción.

Clase `CancelarReservaUseCase` con método: `void Ejecutar(int idReserva)`

- Cancela una reserva existente, cambiando su estado.

Clase `ListarReservasActivasUseCase` con método: `List<Reserva> Ejecutar()`

- Retorna todas las reservas con estado *pendiente de asistencia* o *asistió*.

Clase `ListarActividadesConCupoDisponibleUseCase` con método:
`List<ActividadDeportiva> Ejecutar()`

- Retorna actividades con cupo aún disponible.

No es necesario implementar los usuarios



Permisos de usuario y servicio de autorización

Cada usuario podrá poseer **uno o varios de los siguientes permisos**, que habilitan acciones específicas sobre las entidades del sistema:

Permiso	Descripción
ActividadAlta	Puede crear nuevas actividades deportivas en el centro
ActividadModificacion	Puede modificar los detalles de las actividades deportivas
ActividadBaja	Puede eliminar actividades deportivas del centro
DeporteAlta	Puede dar de alta nuevos deportes que se ofrecen en el centro deportivo
DeporteModificacion	Puede modificar los deportes ya existentes en el sistema
DeporteBaja	Puede eliminar deportes del sistema
InscripcionAlta	Puede registrar nuevas inscripciones de alumnos o miembros a actividades
InscripcionModificacion	Puede modificar las inscripciones de actividades de los alumnos
InscripcionBaja	Puede dar de baja inscripciones a actividades
UsuarioAlta	Puede dar de alta nuevos usuarios del sistema (administradores, entrenadores)
UsuarioModificacion	Puede modificar los datos de los usuarios
UsuarioBaja	Puede dar de baja usuarios del sistema

Servicio de Autorización

Se requiere desarrollar un servicio que **valide si un usuario tiene los permisos necesarios** para realizar una operación sobre el sistema. Este servicio debe implementar la siguiente interfaz:


```
public interface IServicioAutorizacion
{
    bool PoseeElPermiso(int IdUsuario, Permiso permiso);
}
```

Servicio Provisional

Para esta primera entrega, se desarrollará una **implementación provisional** del servicio de autorización llamada `ServicioAutorizacionProvisorio`.

Este servicio responderá de la siguiente manera:

- Si `IdUsuario == 1`, devuelve `true` para cualquier permiso solicitado.
- Si el `IdUsuario` es distinto de `1`, devuelve `false` para todos los permisos.

 **Nota:** Este servicio es **temporal**. En la próxima entrega, se implementará un sistema de gestión de usuarios completo, y el servicio provisional será reemplazado por una versión definitiva que validará los permisos basados en los roles reales asignados a los usuarios.

Resumen:

- **Usuarios** tendrán permisos específicos sobre las actividades, deportes, inscripciones y administración de datos.
- **Autorización:** El sistema de autorización gestionará los permisos mediante un servicio que valida si un usuario tiene acceso para realizar una acción en particular.
- **Servicio provisional:** Durante esta entrega, el servicio de autorización responderá de manera simplificada, y será reemplazado en una futura entrega.

CentroEventos.Repositorios

Implementación de los repositorios utilizando archivos de texto plano. Los repositorios deben manejar el CRUD básico y asegurar persistencia entre ejecuciones.

Importante: Se debe desarrollar un repositorio por cada entidad, esto aporta alta cohesión, menor acoplamiento, mayor testabilidad y mayor escalabilidad.

Nota: Para actualizar los datos en los archivos de texto no se deben tener en cuenta aspectos de rendimiento. Por ejemplo, se podría sobrescribir el archivo completo sólo para modificar un registro en el mismo. En la próxima entrega vamos a utilizar un manejador de bases de datos para gestionar la persistencia de manera más eficiente.

CentroEventos.Consola

Proyecto de consola donde se pueden probar los casos de uso implementados.

En el documento explicativo debe figurar cómo probar funcionalidades desde `Program.cs`.

Requisitos técnicos

- Desarrollar en **.NET 8.0**
- No deshabilitar `<ImplicitUsings>` ni `<Nullable>`
- Resolver todos los **warnings** relacionados con referencias null
- Inyección de dependencias por constructor
- Uso de **interfaces de repositorio**
- Proyecto con estructura limpia y modular