Assessment Técnico — Plataforma de Cursos Online

El objetivo de este assessment es evaluar conocimientos fundamentales en **desarrollo backend con .NET**, consumo de **APIs REST**, modelado de datos, aplicación de **reglas de negocio**, y nociones básicas de arquitectura limpia.

No se espera una solución perfecta ni extremadamente avanzada, sino una implementación **funcional, clara y bien organizada**, priorizando los requisitos obligatorios.

Objetivo General

Desarrollar una **API REST en .NET (8, 9 o 10)** para la gestión de cursos y lecciones, junto con un **frontend desacoplado** que consuma dicha API utilizando tecnologías web estándar o un framework moderno.

La solución debe seguir principios básicos de **Clean Architecture o Hexagonal** y aplicar correctamente las reglas de negocio indicadas, se espera una separación clara de responsabilidades, aunque no necesariamente una implementación avanzada o compleja.

Backend

Tecnologías requeridas

- .NET 8, 9 o 10
- Entity Framework Core
- Base de datos relacional:
  - PostgreSQL o MySQL
- Migraciones con EF Core
- Autenticación con Identity + JWT

Modelo de Datos

Course

- Id (GUID)
- Title (string)
- Status (Draft | Published)
- IsDeleted (bool)
- CreatedAt (DateTime)
- UpdatedAt (DateTime)

Lesson

- Id (GUID)
- CourseId (GUID)
- Title (string)
- Order (int)
- IsDeleted (bool)
- CreatedAt (DateTime)
- UpdatedAt (DateTime)

Relación

- Un Course puede tener muchas Lesson
- Una Lesson pertenece a un solo Course

Frontend (Obligatorio)

El frontend debe consumir la API REST desarrollada.

Puede implementarse utilizando HTML, CSS y JavaScript puro o un framework/librería web como (React, Vue,Angular u otro equivalente)

Funcionalidades requeridas en el frontend

Cursos

- Listar cursos con:
  - Paginación
  - Filtro por estado (Draft / Published)
- Crear curso
- Editar curso
- Eliminar curso (soft delete)
- Publicar / despublicar curso

Lecciones

- Listar lecciones por curso (ordenadas por el campo Order)
- Crear lección
- Editar lección
- Eliminar lección (soft delete)
- Reordenar lecciones (subir/bajar posición)

Autenticación

- Pantalla simple de login
- Almacenar y usar el JWT para llamadas autenticadas

El diseño visual no es prioritario.
Se evaluará principalmente que el frontend funcione correctamente y consuma la API de forma adecuada.

API REST

Endpoints requeridos

POST	/api/auth/register	Sin seguridad
POST	/api/auth/login	Sin seguridad
PATCH	/api/courses/{id}/publish	Con seguridad
PATCH	/api/courses/{id}/unpublish	Con seguridad
GET	/api/courses/search?q=&status=&page=&pageSize=	Con seguridad
GET	/api/courses/{id}/summary	Con seguridad

Reglas de Negocio (Obligatorias)

- Un curso solo puede publicarse si tiene **al menos una lección activa**
- El campo Order de las lecciones debe ser **único dentro del mismo curso**
- La eliminación es lógica (soft delete), no física
- El reordenamiento de lecciones no debe generar órdenes duplicados.
- /publish solo debe permitir publicar cursos que cumplan las reglas de negocio
- /summary debe devolver:
  - Información básica del curso
  - Total de lecciones
  - Fecha de última modificación

Estas reglas deben implementarse en la lógica de negocio, no solo en el frontend.

Base de Datos

- PostgreSQL o MySQL
- Migraciones con EF Core
- Filtro global por IsDeleted
- Seed de:
  - 1 usuario de prueba

Tests

Se deben implementar **al menos 5 tests unitarios** que validen reglas de negocio importantes.

Tests mínimos requeridos

- PublishCourse_WithLessons_ShouldSucceed
- PublishCourse_WithoutLessons_ShouldFail
- CreateLesson_WithUniqueOrder_ShouldSucceed
- CreateLesson_WithDuplicateOrder_ShouldFail
- DeleteCourse_ShouldBeSoftDelete

Los tests deben enfocarse en la lógica de negocio, no en los controladores.

Puntos Extra (Opcionales)

Estos puntos **no son obligatorios** y **no afectan negativamente** la evaluación si no se implementan. Sirven únicamente para destacar.

- Uso de un framework frontend moderno (React, Vue, Angular, etc.)
- Implementación básica de roles (por ejemplo, rol Admin)
- Hard delete solo para usuarios Admin
- Docker (API y base de datos)
- Docker Compose
- Dashboard simple con métricas básicas
- Buen manejo de commits y estructura del repositorio

Orden de Prioridades

1. Cumplir **todos los requisitos obligatorios**
2. Asegurar que la solución funcione correctamente
3. Implementar tests mínimos
4. Agregar puntos extra solo si queda tiempo disponible

Es preferible una solución simple y completa que una solución incompleta con muchas características opcionales.

Entregables

Repositorio o repositorios Git que contenga:

- Solución .sln
- Proyectos separados por capas
- Backend (API)
- Frontend (en carpeta separada o repositorio aparte)
- Archivo README.md con:
  - Pasos para configurar la base de datos
  - Comandos para ejecutar migraciones
  - Cómo levantar la API y el frontend
  - Credenciales del usuario de prueba