**Caso Práctico: Gestión de Matrículas**

**Requisitos Funcionales**

**Gestión de Matrículas:**

* Crear una nueva matrícula (asociada a un estudiante y un curso).
* Modificar el estado de una matrícula (ej: "Activa", "Cancelada", "Finalizada").
* Eliminar una matrícula (solo si está en estado "Cancelada").
* Consultar matrículas por:
* ID de matrícula.
* ID de estudiante.
* ID de curso.
* Estado.
* Validar que un estudiante no se matricule dos veces en el mismo curso.

**Desarrollo Técnico**

1. **Configuración del Proyecto**

**Tecnologías:**

* ASP.NET Core Web API (.NET 6.0 o posterior).
* Entity Framework Core / Store Procedure (utilizar ambos).
* SQL Server 2012 posterior.
* Swagger/OpenAPI para documentación.

1. **Validaciones Específicas**

**Creación de Matrícula:**

* El estudiante y el curso deben existir.
* No permitir matrícula duplicada (mismo StudentId y CourseId).
* EnrollmentDate no puede ser futura.

**Actualización:**

* Solo se permite cambiar el Status.
* No permitir cambiar a "Cancelada" si ya está "Finalizada".

**Eliminación:**

* Solo si el estado es "Cancelada".

**Evaluación Esperada**

**Criterios de Éxito:**

* Proponer estructura de proyecto y aplicar buenas prácticas.
* Correcta implementación de los endpoints CRUD.
* Validaciones sólidas (ej: duplicados, estados).
* Uso de Entity Framework Core (relaciones, migraciones).
* Manejo de errores y respuestas HTTP claras.
* Documentación en Swagger.

**Bonus:**

* Implementar DTOs y AutoMapper.
* Incluir logging (ej: Serilog).
* Usar patrones como Repository o Unit of Work.
* Utilizar procedimientos almacenados

**Manera de Entrega de Proyecto:**

* En Repositorio GIT
* Enlace Drive Comprimido (zip, rar)

**Casos de pruebas Funcionales**

Nombre: Joel Isaias Miguel Hernández Livia

Fecha: 01/06/2025

Ambiente: local

Herramientas: Swagger, SQL Server

|  |  |  |
| --- | --- | --- |
| **Descripción** | **Resultado esperado** | **Estado** |
| Matricular cuando el estudiante y el curso existan | HTTP Response: 200 | Pasó |
| Intentar matricular cuando el estudiante exista, pero el curso no | HTTP Response: 400 | Pasó |
| Intentar matricular cuando el estudiante no exista, pero el curso sí | HTTP Response: 400 | Pasó |
| Intentar matricular 2 veces al mismo estudiante en el mismo curso | HTTP Response: 400 | Pasó |
| Actualizar el estatus: de “Activa” a “Cancelada” | HTTP Response: 200 | Pasó |
| Actualizar el estatus: de “Cancelada” a “Finalizada” | HTTP Response: 200 | Pasó |
| Intentar actualizar el estatus de “Finalizada” a “Cancelada” | HTTP Response: 400 | Pasó |
| Consultar matricula por ID de matrícula | Datos de la matricula | Pasó |
| Consultar matricula por ID de estudiante | Datos de la matricula | Pasó |
| Consultar matricula por ID de curso | Datos de la matricula | Pasó |
| Consultar matricula por estado | Datos de la matricula | Pasó |
| Comprobar la fecha de registro | No puede ser futura | Pasó |
| Comprobar que solo se permita cambiar el estatus | La API no debe tener opción para actualizar otros campos | Pasó |
| Eliminar matricula con estado “Cancelada” | HTTP Response: 200 | Pasó |
| Intentar eliminar matricula con estado “Activa” | HTTP Response: 400 | Pasó |
| Intentar eliminar matricula con estado “Finalizada” | HTTP Response: 400 | Pasó |