**Trazabilidad del flujo al crear un Estudiante (Student)**

Aquí se explica paso a paso todo el flujo que se sigue desde que un cliente (por ejemplo, Swagger) envía una solicitud para crear un estudiante, hasta que se guarda en la base de datos y se retorna el resultado:

**1. Inicio: Llamada HTTP desde el cliente (POST /api/Student)**

Cuando el usuario usa Swagger o Postman y hace un **POST** a /api/Student, con un JSON como:

{

"id": 0,

"lastName": "Pérez",

"firstMidName": "Juan",

"enrollmentDate": "2025-07-24"

}

→ Esta solicitud llega al siguiente método del controlador:

[HttpPost]

public async Task<ActionResult<StudentDTO>> Create(StudentDTO studentDTO)

{

var student = \_mapper.Map<Student>(studentDTO);

var newStudent = await \_studentService.Insert(student);

return Ok(\_mapper.Map<StudentDTO>(newStudent));

}

**2. Mapeo del DTO a entidad**

Usando **AutoMapper**, el DTO (StudentDTO) es convertido a una entidad (Student):

var student = \_mapper.Map<Student>(studentDTO);

Esto usa la configuración:

CreateMap<Student, StudentDTO>().ReverseMap();

**3. Llamada al servicio de dominio (StudentService)**

La línea:

var newStudent = await \_studentService.Insert(student);

Llama al método Insert definido en:

public interface IStudentService : IGenericService<Student> { }

public class StudentService : GenericService<Student>, IStudentService

{

public StudentService(IStudentRepository repo) : base(repo) { }

}

Y este hereda la lógica del método Insert en GenericService<TEntity>:

**4. Llamada al repositorio (StudentRepository)**

En el GenericService:

public Task<TEntity> Insert(TEntity entity) => \_repository.Insert(entity);

Esto llama a:

public interface IStudentRepository : IGenericRepository<Student> { }

public class StudentRepository : GenericRepopsitory<Student>, IStudentRepository

{

public StudentRepository(SchoolContext context) : base(context) { }

}

→ Y como hereda de GenericRepopsitory<Student>, entra en:

**5. Guardado en la base de datos con EF Core**

public async Task<TEntity> Insert(TEntity entity)

{

await \_context.Set<TEntity>().AddAsync(entity); // <-- Aquí se agrega a la memoria del DbContext

await \_context.SaveChangesAsync(); // <-- Aquí se persiste en la base de datos SQL Server

return entity; // <-- Se retorna la entidad con su ID generado

}

* Aquí TEntity es Student.
* El DbContext (SchoolContext) ya está configurado con la cadena de conexión y conoce la tabla students.

**6. Regreso de la entidad creada**

El Student insertado sube por la misma cadena de capas:

* GenericRepopsitory → StudentRepository → StudentService → StudentController.

**7. Mapeo final a DTO y retorno al cliente**

En el StudentController:

return Ok(\_mapper.Map<StudentDTO>(newStudent));

Esto convierte el Student a StudentDTO (para evitar exponer datos innecesarios) y lo devuelve en formato JSON al cliente.

**Tabla 1.**

*Resumen de la traza*

| **Nivel** | **Clase/Archivo** | **Método llamado** | **Rol** |
| --- | --- | --- | --- |
| Cliente (Swagger) | POST /api/Student |  | Envía JSON con datos del estudiante |
| Controlador | StudentController | Create(StudentDTO studentDTO) | Recibe y convierte a entidad |
| Servicio | StudentService (vía GenericService) | Insert(Student student) | Encapsula la lógica de negocio |
| Repositorio | StudentRepository (vía GenericRepopsitory) | Insert(Student student) | Interactúa con EF Core |
| EF Core | SchoolContext | AddAsync y SaveChangesAsync | Guarda el estudiante en la BD |
| Retorno | JSON | DTO mapeado | Devuelve el resultado al cliente |

**Resultado final**

* Se guarda el estudiante en la tabla students de la base de datos ContosoUniversity.
* Swagger devuelve el estudiante creado con su ID asignado.