

Asignatura	Datos del alumno	Fecha
Ciberinfraestructura	Apellidos: Aguilar Islas	17/05/2025
	Nombre: Josué Norberto	

Actividades

Trabajo: Generación de un web service

Descripción de la actividad

Los servicios web nos ayudan a generar canales de comunicación ágiles para módulos de desarrollo pequeños. Con un web service podemos, por ejemplo, realizar operaciones CRUD (Create, Read, Update, Delete) sobre una base de datos.

REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Un web service basado en REST nos devolverá una respuesta, seguida del estatus del servicio, en uno de estos formatos, según se haya configurado.

Ejemplo de formato:

XML	JSON
<pre><?xml version="1.0" encoding="UTF-8" ?> <root> <student> <id>01</id> <name>Tom</name> <lastname>Price</lastname> </student> <student> <id>02</id> <name>Nick</name> <lastname>Thameson</lastname> </student> </root></pre>	<pre>{ "student": [{ "id": "01", "name": "Tom", "lastname": "Price" }, { "id": "02", "name": "Nick", "lastname": "Thameson" }] }</pre>

Objetivos

Crear un web service que genere las operaciones:

GET	Sobre el listado completo de registros en base de datos
GET Parametrizado	Con el identificador de un registro, para obtener el valor del mismo exclusivamente
POST	Insertar un registro nuevo

Las operaciones deberán retornar una respuesta en JSON.

Códigos de respuesta HTTP:

HTTP Status Codes

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone
304	Not Modified	500	Internal Server Error
307	Temporary Redirect	503	Service Unavailable

Desarrollo.

El presente trabajo muestra la implementación de un Web Service RESTful utilizando .NET y PostgreSQL. Se implementaron tres operaciones principales (GET, GET con parámetro y POST) para la gestión de registros de estudiantes. Las respuestas se generan en formato JSON utilizando la arquitectura de ASP.NET Core Web API.

El proyecto se organizó en carpetas para Controllers, Models y Data. Se empleó Entity Framework Core para manejar la base de datos PostgreSQL y se utilizó Swagger para la prueba de los endpoints.

Endpoints.

- GET /api/students

Este endpoint devuelve la lista completa de estudiantes registrados.

Responses


Curl

```
curl -X 'GET' \
'http://localhost:5052/api/Student' \
-H 'accept: text/plain'
```

Request URL

```
http://localhost:5052/api/Student
```

Server response

Code	Details
200	<div><h6>Response body</h6><pre>[]</pre><div> Download</div></div> <div><h6>Response headers</h6><pre>content-type: application/json; charset=utf-8 date: Sun, 18 May 2025 05:58:38 GMT server: Kestrel transfer-encoding: chunked</pre></div>

Responses

Code	Description	Links
200	OK	No links

Media type

text/plain

▼

Controls Accept header.

Example Value | Schema

```
[
  {
    "id": 0,
    "name": "string",
    "lastname": "string"
  }
]
```

- **GET /api/students/{id}**

Este endpoint devuelve un estudiante específico mediante su identificador.

POST /api/Student

Parameters

Try it out

No parameters

Request body

application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "lastname": "string"
}
```

Responses

Code	Description	Links
200	OK <div>Media type<div>text/plain</div><div>Controls Accept header.</div><div>Example Value Schema<div><pre>{ "id": 0, "name": "string", "lastname": "string" }</pre></div></div></div>	No links

- POST /api/students

Este endpoint permite insertar un nuevo estudiante a la base de datos. Requiere los campos Name y EnrollmentCode.

Parameters

Try it out

Name	Description
id * required	
integer(\$int32)	1
(path)	

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5052/api/Student/1' \
  -H 'accept: text/plain'
```

Request URL

http://localhost:5052/api/Student/1

Server response

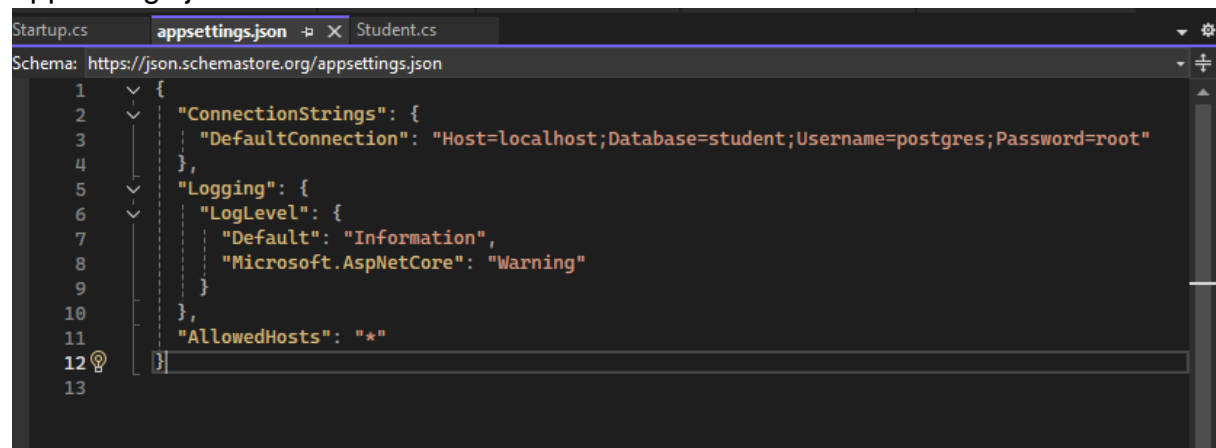
Code	Details
200	<div><div>Response body</div><pre>{ "id": 1, "name": "string", "lastname": "string" }</pre><div>Download</div></div> <div><div>Response headers</div><pre>content-type: application/json; charset=utf-8 date: Sun, 18 May 2025 05:58:52 GMT server: Kestrel transfer-encoding: chunked</pre></div>

Responses

Code	Description	Links
------	-------------	-------

Evidencia de configuración

appsettings.json



```
1 {
2   "ConnectionStrings": {
3     "DefaultConnection": "Host=localhost;Database=student;Username=postgres;Password=root"
4   },
5   "Logging": {
6     "LogLevel": {
7       "Default": "Information",
8       "Microsoft.AspNetCore": "Warning"
9     }
10  },
11  "AllowedHosts": "*"
12 }
13
```

Startup.cs

```
Startup.cs*  X appsettings.json  Student.cs
CrudStudentAPI  CrudStudentAPI.Startup  Configure(IApplicationBuilder app, IWebHostEnvironment env)

1  using Microsoft.Extensions.Options;
2  using CrudStudentAPI.Data;
3  using Microsoft.EntityFrameworkCore;
4  using Microsoft.OpenApi.Models;
5
6  namespace CrudStudentAPI
7  {
8      2 references
9      public class Startup
10     {
11         2 references
12         public IConfiguration configuration { get; }
13         0 references
14         public Startup(IConfiguration configuration)
15         {
16             this.configuration = configuration;
17         }
18         0 references
19         public void ConfigureServices(IServiceCollection services)
20         {
21             services.AddControllers();
22             services.AddDbContext<AppDbContext>(options => options.UseNpgsql(configuration.GetConnectionString("DefaultConnection")));
23             services.AddScoped<IStudentRepository, StudentRepository>();
24
25             // Add Swagger services
26             services.AddSwaggerGen();
27         }
28
29         0 references
30         public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
31         {
32             if (env.IsDevelopment())
33             {
34                 app.UseDeveloperExceptionPage();
35                 // Enable Swagger in development
36                 app.UseSwagger();
37                 app.UseSwaggerUI(c =>
38                 {
39                     c.SwaggerEndpoint("/swagger/v1/swagger.json", "CrudStudentAPI v1");
40                     c.RoutePrefix = string.Empty; // Set Swagger UI at the app's root
41                 });
42             }
43             app.UseRouting();
44             app.UseEndpoints(endpoints =>
45             {
46                 endpoints.MapControllers();
47             });
48         }
49     }
50 }
```

Conclusión

La implementación y prueba de este servicio permitió aplicar conceptos de arquitectura RESTful, consumo de bases de datos mediante Entity Framework Core, y validación de operaciones HTTP básicas. El sistema quedó listo para extenderse a futuras operaciones CRUD completas.