Facultad de Ingeniería

Escuela de Computación

Desarrollo de Software Empresarial

Guía 8:Redis

PRESENTADO POR:

Nombre	Código
Eduardo Josué Deras Mancia	DM201736

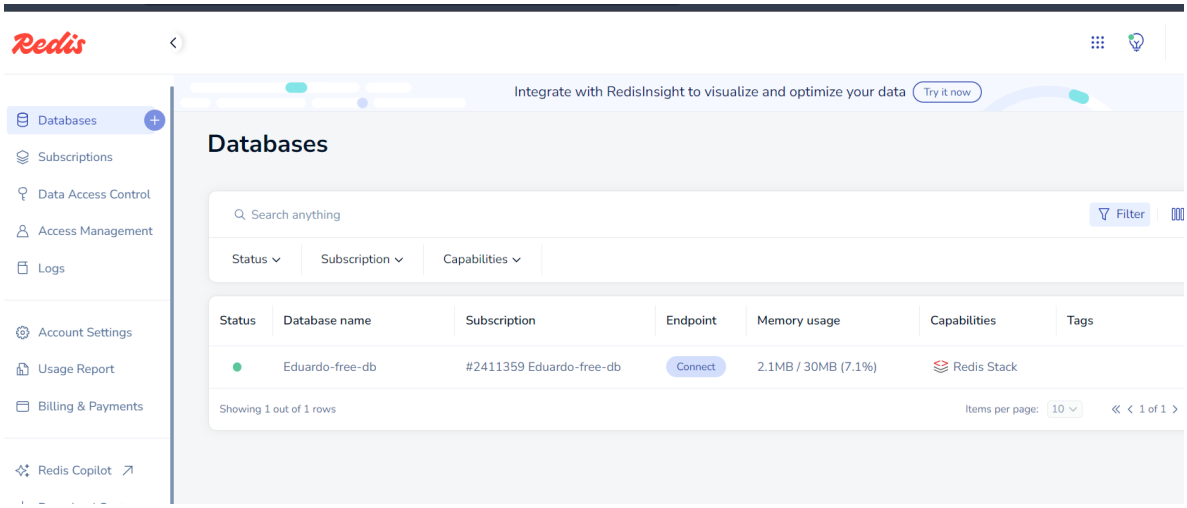
Repositorio del procedimiento y desarrollo de habilidades

guía 8: https://github.com/EduardMancia48/guia8-des

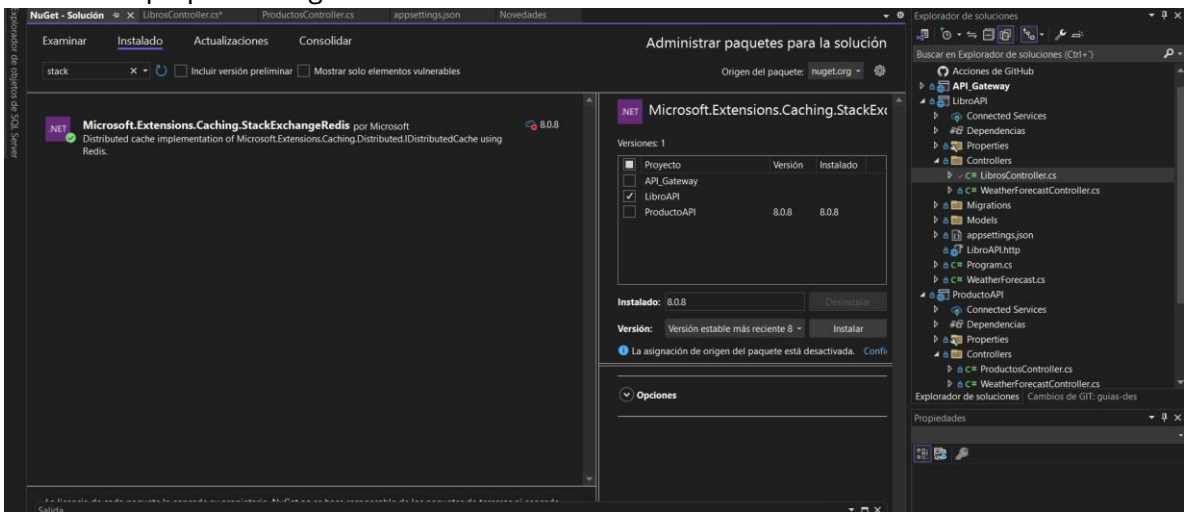
Catedrático: Mg. Emerson Cartagena

Desarrollo de habilidades

1. Creación de cuenta en Redis



2. Instalando paquetes Nuget

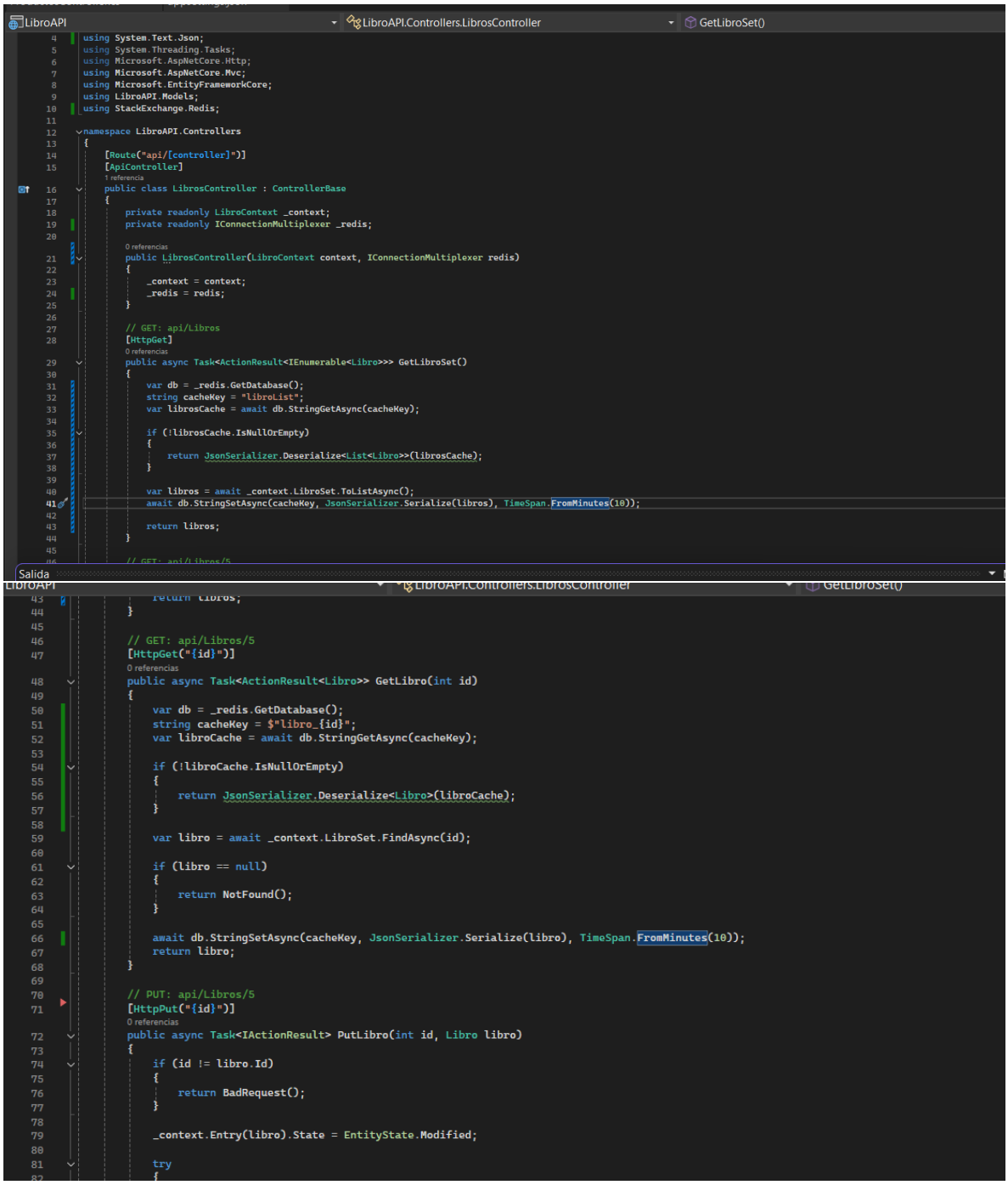


3. Configurando appsettings.json

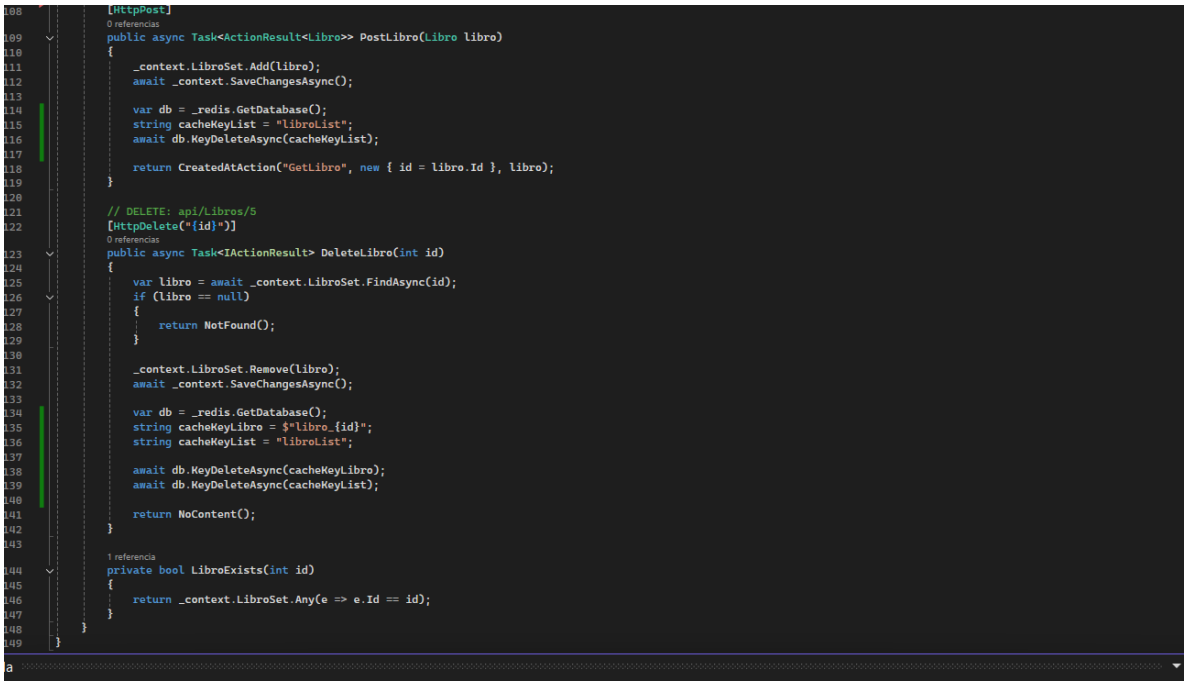
4. Configurando program.cs:

```
builder.Services.AddDbContext<LibroContext>(options =>
options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Add services to the container.
builder.Services.AddStackExchangeRedisCache(opciones =>
{
    opciones.Configuration = builder.Configuration.GetConnectionString("redis");
});

//Registrar IconnectionMultiplexer
builder.Services.AddSingleton<IConnectionMultiplexer>(sp =>
{
    var configuration = ConfigurationOptions.Parse(builder.Configuration.GetConnectionString("redis"), true);
    return ConnectionMultiplexer.Connect(configuration);
});
builder.Services.AddOutputCache();

builder.Services.AddControllers();
```

5. Codigo de LibroController.cs




```
using System.Text.Json;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using LibroAPI.Models;
using StackExchange.Redis;

namespace LibroAPI.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    1 referencia
    public class LibrosController : ControllerBase
    {
        private readonly LibroContext _context;
        private readonly IConnectionMultiplexer _redis;

        0 referencias
        public LibrosController(LibroContext context, IConnectionMultiplexer redis)
        {
            _context = context;
            _redis = redis;
        }

        // GET: api/Libros
        [HttpGet]
        0 referencias
        public async Task<ActionResult<IEnumerable<Libro>>> GetLibroSet()
        {
            var db = _redis.GetDatabase();
            string cacheKey = "libroList";
            var librosCache = await db.StringGetAsync(cacheKey);

            if (!librosCache.IsNullOrEmpty)
            {
                return JsonSerializer.Deserialize<List<Libro>>(librosCache);
            }

            var libros = await _context.LibroSet.ToListAsync();
            await db.StringSetAsync(cacheKey, JsonSerializer.Serialize(libros), TimeSpan.FromMinutes(10));

            return libros;
        }

        // GET: api/Libros/5
        [HttpGet("{id}")]
        0 referencias
        public async Task<ActionResult<Libro>> GetLibro(int id)
        {
            var db = _redis.GetDatabase();
            string cacheKey = $"libro_{id}";
            var libroCache = await db.StringGetAsync(cacheKey);

            if (!libroCache.IsNullOrEmpty)
            {
                return JsonSerializer.Deserialize<Libro>(libroCache);
            }

            var libro = await _context.LibroSet.FindAsync(id);

            if (libro == null)
            {
                return NotFound();
            }

            await db.StringSetAsync(cacheKey, JsonSerializer.Serialize(libro), TimeSpan.FromMinutes(10));
            return libro;
        }

        // PUT: api/Libros/5
        [HttpPut("{id}")]
        0 referencias
        public async Task<IActionResult> PutLibro(int id, Libro libro)
        {
            if (id != libro.Id)
            {
                return BadRequest();
            }

            _context.Entry(libro).State = EntityState.Modified;

            try
            {
```



```
        [HttpPost]
        0 referencias
        public async Task<ActionResult<Libro>> PostLibro(Libro libro)
        {
            _context.LibroSet.Add(libro);
            await _context.SaveChangesAsync();

            var db = _redis.GetDatabase();
            string cacheKeyList = "libroList";
            await db.KeyDeleteAsync(cacheKeyList);

            return CreatedAtAction("GetLibro", new { id = libro.Id }, libro);
        }

        // DELETE: api/Libros/5
        [HttpDelete("{id}")]
        0 referencias
        public async Task<IActionResult> DeleteLibro(int id)
        {
            var libro = await _context.LibroSet.FindAsync(id);
            if (libro == null)
            {
                return NotFound();
            }

            _context.LibroSet.Remove(libro);
            await _context.SaveChangesAsync();

            var db = _redis.GetDatabase();
            string cacheKeyLibro = $"libro_{id}";
            string cacheKeyList = "libroList";

            await db.KeyDeleteAsync(cacheKeyLibro);
            await db.KeyDeleteAsync(cacheKeyList);

            return NoContent();
        }

        1 referencia
        private bool LibroExists(int id)
        {
            return _context.LibroSet.Any(e => e.Id == id);
        }
    }
}
```

a