Hypermedia-Demo: End-to-End .NET 9 Web API with Manual HAL, JSON-LD, Siren & Pagination

A **comprehensive, step-by-step** guide to build an ASP.NET Core 9 Web API supporting **HATEOAS** via **manual HAL**, **JSON-LD**, and **Siren**—complete with dynamic links, pagination, folder structure, interfaces, and Postman testing. We'll also compare **with** and **without** HAL.

Prerequisites

- .NET 9 SDK: https://dotnet.microsoft.com/download/dotnet/9.0
- Code editor: Visual Studio 2022 / VS Code
- Postman (or similar REST client)
- Terminal (bash or PowerShell)

---

1. Create Solution & Project

Create main folder and enter it:

```
mkdir HypermediaDemo
cd HypermediaDemo
```

1.

Create Web API project:

```
dotnet new webapi -n HypermediaDemo.Api --framework net9.0
```

2.

Create solution file (same level):

```
dotnet new sln -n HypermediaDemo
```

3.

Add project to solution:

dotnet sln HypermediaDemo.sln add HypermediaDemo.Api/HypermediaDemo.Api.csproj

4.

Ensure you run these commands from the `HypermediaDemo` root.

---

2. Add NuGet Packages

Switch into the API project:

cd HypermediaDemo.Api

Install packages:

```
# Enable Newtonsoft.Json support
dotnet add package Microsoft.AspNetCore.Mvc.NewtonsoftJson
dotnet add package Newtonsoft.Json

# JSON-LD and Siren support
dotnet add package LinkedData.JsonLd
dotnet add package FluentSiren
```

> **Note:** We implement HAL links manually—no HAL NuGet package is needed.

---

3. Organize Folder Structure

Under `HypermediaDemo.Api/`, create:

```
Models/
Interfaces/
Repositories/
Formatters/
```

---

4. Define Interfaces & Repositories

4.1 `IProductRepository` (Interfaces/IProductRepository.cs)

```
using System.Collections.Generic;
using HypermediaDemo.Api.Models;

namespace HypermediaDemo.Api.Interfaces
{
    public interface IProductRepository
    {
        IEnumerable<Product> GetAll();
        Product? GetById(int id);
    }
}
```

4.2 `ProductRepository` (Repositories/ProductRepository.cs)

```
using System.Collections.Generic;
using System.Linq;
using HypermediaDemo.Api.Interfaces;
using HypermediaDemo.Api.Models;

namespace HypermediaDemo.Api.Repositories
{
    public class ProductRepository : IProductRepository
    {
        private readonly List<Product> _store = Enumerable.Range(1, 50)
            .Select(i => new Product { Id = i, Name = $"Product {i}" })
            .ToList();

        public IEnumerable<Product> GetAll() => _store;
        public Product? GetById(int id) => _store.SingleOrDefault(p => p.Id == id);
    }
}
```

---

5. Create Models

5.1 `Resource` base class (Models/Resource.cs)

```
using System.Collections.Generic;
using Newtonsoft.Json;

namespace HypermediaDemo.Api.Models
```

```
{
    public abstract class Resource
    {
        [JsonProperty("_links")]
        public IDictionary<string, object> Links { get; set; } = new Dictionary<string, object>();
    }
}
```

5.2 `Product` model (Models/Product.cs)

```
using Newtonsoft.Json;

namespace HypermediaDemo.Api.Models
{
    public class Product : Resource
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;

        // JSON-LD metadata
        [JsonProperty("@context", NullValueHandling = NullValueHandling.Ignore)]
        public string Context => "https://schema.org/";

        [JsonProperty("@id", NullValueHandling = NullValueHandling.Ignore)]
        public string? IdLink { get; set; }

        [JsonProperty("@type", NullValueHandling = NullValueHandling.Ignore)]
        public string Type => "Product";
    }
}
```

---

6. Implement Siren Formatter

File: `Formatters/SirenOutputFormatter.cs`

```
using System;
using System.Text;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc.Formatters;
using Microsoft.Net.Http.Headers;
using Newtonsoft.Json;
```

```
namespace HypermediaDemo.Api.Formatters
{
    public class SirenOutputFormatter : TextOutputFormatter
    {
        public SirenOutputFormatter()
        {

SupportedMediaTypes.Add(MediaTypeHeaderValue.Parse("application/vnd.siren+json"));
            SupportedEncodings.Add(Encoding.UTF8);
        }

        protected override bool CanWriteType(Type? type) => true;

        public override async Task WriteResponseBodyAsync(OutputFormatterWriteContext
context, Encoding encoding)
        {
            var json = JsonConvert.SerializeObject(context.Object, Formatting.Indented);
            await context.HttpContext.Response.WriteAsync(json, encoding);
        }
    }
}
```

---

7. Configure `Program.cs`

File: `Program.cs`

```
using HypermediaDemo.Api.Formatters;
using HypermediaDemo.Api.Interfaces;
using HypermediaDemo.Api.Repositories;
using Microsoft.AspNetCore.Mvc;

var builder = WebApplication.CreateBuilder(args);

// Add controllers, enable Newtonsoft JSON, and register Siren formatter
builder.Services.AddControllers(options =>
{
    options.OutputFormatters.Add(new SirenOutputFormatter());
    options.FormatterMappings.SetMediaTypeMappingForFormat("hal",    "application/hal+json");
    options.FormatterMappings.SetMediaTypeMappingForFormat("ldjson","application/ld+json");
```

```
options.FormatterMappings.SetMediaTypeMappingForFormat("siren","application/vnd.siren+jso
n");
})
.AddNewtonsoftJson();

// Register repository
builder.Services.AddSingleton<IProductRepository, ProductRepository>();

var app = builder.Build();
app.MapControllers();
app.Run();
```

> **Key:** No `AddHalSupport()`—HAL links are built manually in controllers.

---

8. Implement Controllers

8.1 HAL Controller (with pagination)

File: `Controllers/ProductsHalController.cs`

```
using System.Linq;
using System.Collections.Generic;
using HypermediaDemo.Api.Interfaces;
using HypermediaDemo.Api.Models;
using Microsoft.AspNetCore.Mvc;

namespace HypermediaDemo.Api.Controllers
{
    [ApiController]
    [Route("api/hal/products")]
    public class ProductsHalController : ControllerBase
    {
        private readonly IProductRepository _repo;
        public ProductsHalController(IProductRepository repo) => _repo = repo;

        [HttpGet("{id}", Name = "GetHalProduct")]
        [Produces("application/hal+json")]
        public IActionResult Get(int id)
        {
            var p = _repo.GetById(id);
```

```
        if (p == null) return NotFound();

        // Build HAL links manually
        p.Links["self"] = new { href = Url.Link("GetHalProduct", new { id }) };
        p.Links["all"]  = new { href = Url.Link("GetHalProducts", new { page = 1, size = 10 }) };
        return Ok(p);
    }

    [HttpGet(Name = "GetHalProducts")]
    [Produces("application/hal+json")]
    public IActionResult GetAll(int page = 1, int size = 10)
    {
        var all = _repo.GetAll().ToList();
        var items = all.Skip((page - 1) * size).Take(size);
        var total = all.Count;

        var result = new
        {
            count = total,
            page,
            size,
            _links = new Dictionary<string, object>
            {
                ["self"] = new { href = Url.Link("GetHalProducts", new { page, size }) },
                ["next"] = new { href = Url.Link("GetHalProducts", new { page = page + 1, size }) },
                ["prev"] = new { href = Url.Link("GetHalProducts", new { page = page > 1 ? page - 1
: 1, size }) }
            },
            _embedded = new
            {
                products = items.Select(p => new
                {
                    p.Id,
                    p.Name,
                    _links = new { self = new { href = Url.Link("GetHalProduct", new { id = p.Id }) } }
                })
            }
        };
        return Ok(result);
    }
  }
}
```

8.2 JSON-LD Controller

File: `Controllers/ProductsJsonLdController.cs`

```
using HypermediaDemo.Api.Interfaces;
using Microsoft.AspNetCore.Mvc;

namespace HypermediaDemo.Api.Controllers
{
    [ApiController]
    [Route("api/ld/products")]
    public class ProductsJsonLdController : ControllerBase
    {
        private readonly IProductRepository _repo;
        public ProductsJsonLdController(IProductRepository repo) => _repo = repo;

        [HttpGet("{id}", Name = "GetJsonLdProduct")]
        [Produces("application/ld+json")]
        public IActionResult Get(int id)
        {
            var p = _repo.GetById(id);
            if (p == null) return NotFound();

            // Populate JSON-LD @id
            p.IdLink = Url.Link("GetJsonLdProduct", new { id });
            return Ok(p);
        }
    }
}
```

8.3 Siren Controller

File: `Controllers/ProductsSirenController.cs`

```
using System.Linq;
using System.Collections.Generic;
using HypermediaDemo.Api.Interfaces;
using Microsoft.AspNetCore.Mvc;

namespace HypermediaDemo.Api.Controllers
{
    [ApiController]
    [Route("api/siren/products")]
```

```
    public class ProductsSirenController : ControllerBase
    {
        private readonly IProductRepository _repo;
        public ProductsSirenController(IProductRepository repo) => _repo = repo;

        [HttpGet("{id}", Name = "GetSirenProduct")]
        [Produces("application/vnd.siren+json")]
        public IActionResult Get(int id)
        {
            var p = _repo.GetById(id);
            if (p == null) return NotFound();

            var entity = new
            {
                @class = new[] { "product" },
                properties = new { p.Id, p.Name },
                links = new[]
                {
                    new { rel = new[] { "self" }, href = Url.Link("GetSirenProduct", new { id }) },
                    new { rel = new[] { "collection" }, href = Url.Link("GetSirenProductsPaged", new {
page = 1, size = 5 }) }
                }
            };
            return Ok(entity);
        }

        [HttpGet(Name = "GetSirenProductsPaged")]
        [Produces("application/vnd.siren+json")]
        public IActionResult GetAll(int page = 1, int size = 5)
        {
            var all = _repo.GetAll().ToList();
            var items = all.Skip((page - 1) * size).Take(size);

            var links = new List<object>
            {
                new { rel = new[] { "self" }, href = Url.Link("GetSirenProductsPaged", new { page, size
}) }
            };
            if (page * size < all.Count)
                links.Add(new { rel = new[] { "next" }, href = Url.Link("GetSirenProductsPaged", new {
page = page + 1, size }) });

            var siren = new
            {
```

```
            @class = new[] { "product", "collection" },
            entities = items.Select(p => new
            {
                @class = new[] { "product" },
                rel = new[] { "item" },
                properties = new { p.Id, p.Name },
                links = new[] { new { rel = new[] { "self" }, href = Url.Link("GetSirenProduct", new { id
= p.Id }) } }
            }),
            links
        };
        return Ok(siren);
    }
  }
}
```

---

9. Compare HAL vs. Without HAL

Aspect	Without HAL	With HAL (`application/hal+json`)
Response body	`{ "Id":1, "Name":"P1" }`	`{ "Id":1, "Name":"P1", _links:{...} }`
Discoverability	Client hardcodes route knowledge	Clients follow `_links.self`, `next`
Evolution	Breaking changes if routes change	`Url.Link()` keeps links up-to-date
Pagination	No built-in navigation links	`_links.next` & `_links.prev` present

---

10. Run & Test with Postman

Build & run:

```
 dotnet build
dotnet run
```

1.
2. **Set up Postman collections**:

- **HAL**:
  - GET `https://localhost:5001/api/hal/products/1` → Header: `Accept: application/hal+json`
  - GET `.../api/hal/products?page=2&size=5` → `Accept: application/hal+json`
- **JSON-LD**:
  - GET `https://localhost:5001/api/ld/products/1` → `Accept: application/ld+json`
- **Siren**:
  - GET `https://localhost:5001/api/siren/products/1` → `Accept: application/vnd.siren+json`
  - GET `.../api/siren/products?page=2&size=3` → `Accept: application/vnd.siren+json`

3. **Observe**:
   - **Raw JSON** (no Accept) vs. **HAL** shows `_links` injection
   - JSON-LD includes `@context`, `@id`, `@type`
   - Siren responses have `class`, `properties`, `entities`, `links`

You now have a fully working .NET 9 Web API showcasing **manual HAL**, **JSON-LD**, and **Siren** hypermedia, with pagination and clear Postman test instructions.