



Kauno technologijos universitetas
Informatikos fakultetas

Architektūros specifikacija

Tadas Laurinaitis
studentas

Doc. Šarūnas Packevičius
Dėstytojas

Kaunas, 2024

Contents

1. Komponento aprašymas	3
1.1. Teikiamos funkcijos	3
1.2. Paskirtis	3
1.3. Kokias problemas sprendžia	3
1.4. Kokiu tikslu galima naudoti	3
1.4. Kokie apribojimai taikomi komponento naudojimui	3
1.5. Kita svarbi informacija komponento vartojimui	3
2. Komponento architektūra	4
2.1. Klasių diagrama	4
2.2. Sekų diagrama	4
2.3. Būsenų diagrama	5
2.4. Išdėstymo diagrama	5
2.5. Kita papildoma informacija ir modeliai, kuriais remiantis galima būtų suprasti kaip komponentas funkcionuoja	5
3. Komponento aprašymas	6
3.1. Klasių diagramos	6
3.2. Detalūs metodų, klasių, paketų aprašai	6
3.3. Dokumentacija	7
3.4. Naudojimo pavyzdys	9
3.5. Komponento vartojimo vadovas	9
3.6. Komponento diegimo instrukcijos ir papildomi reikalavimai naudojimui	10
3.7. Reikalavimai diegimo aplinkai	10
3.8. Pakeitimų istorija	11
3.9. Licencija	11
4. Rinkoje esamų komponentų apžvalga	12

1. Komponento aprašymas

1.1. Teikiamos funkcijos

Komponentas suteikia galimybę naudotis bendros paskirties HTTP kliento sąsaja (angl. Generic HTTP Client), kuri palaiko GET, POST, PUT, DELETE užklausų tipus, turi automatinę JSON atsakymų deserializaciją bei išsamų klaidų tvarkymą, su aiškiais atsakymų statusais ir klaidų pranešimais.

1.2. Paskirtis

Šio komponento paskirtis yra suteikti naudotojui galimybę kurti naujas API klientines aplikacijas greičiau ir lengviau, negalvojant apie daug laiko užimančius procesus, tokius kaip - objektų deserializacija, klaidų tvarkymas ir klaidų pranešimų apdorojimas.

1.3. Kokias problemas sprendžia

Kuriant API klientines aplikacijas dažnu atveju būna labai daug pasikartojančio kodo, ypač susijusio su klaidų valdymu ir JSON deserializacija, kuris galiausiai apkrauna visą kuriamą sistemos architektūrą ir apsunkina navigaciją tos pačios sistemos kode. Šis komponentas leidžia naudotojui išvengti šios problemos, padengdamas ją sukeliančius veiksnius bendros paskirties kodu, kurį lengva naudoti - paduodant url kartu su norimais objektais į komponentą.

1.4. Kokiu tikslu galima naudoti

Šį komponentą galima naudoti aplikacijose, dirbančiuose su HTTP protokolu grįžtais API, klientinės logikos rašymo palengvinimui, kadangi padengiama tiek objektų deserializacijos, tiek klaidų valdymo ir pranešimų grąžinimo dalys.

1.4. Kokie apribojimai taikomi komponento naudojimui

Komponentas neveikia su senesnėmis nei .NET 8 versijomis. Komponentas skirtas darbui tik su HTTP protokolą naudojančiais API.

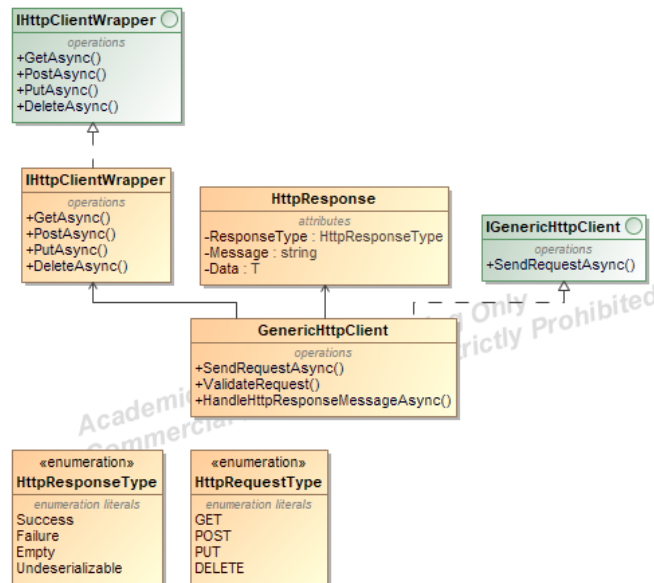
1.5. Kita svarbi informacija komponento vartojimui

Komponentas yra pritaikytas naudoti .NET projektuose, turinčiuose bent .NET 8 versiją.

2. Komponento architektūra

2.1. Klasių diagrama

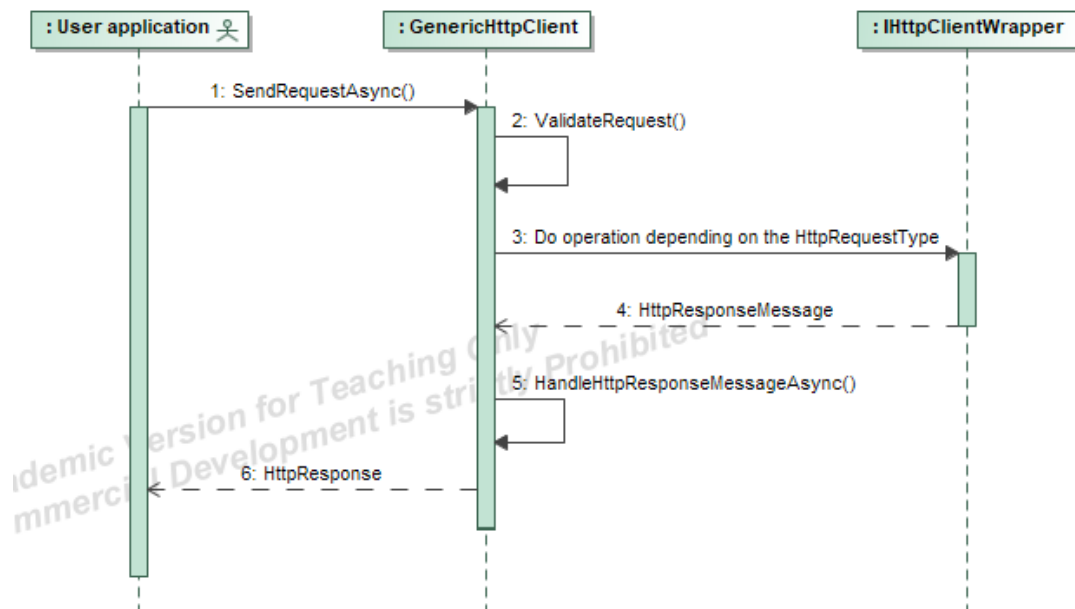
Komponento klasių diagrama pateikiama žemiau esančiame paveikslėlyje (1 pav.).



1 pav. Komponento klasių diagrama.

2.2. Sekų diagrama

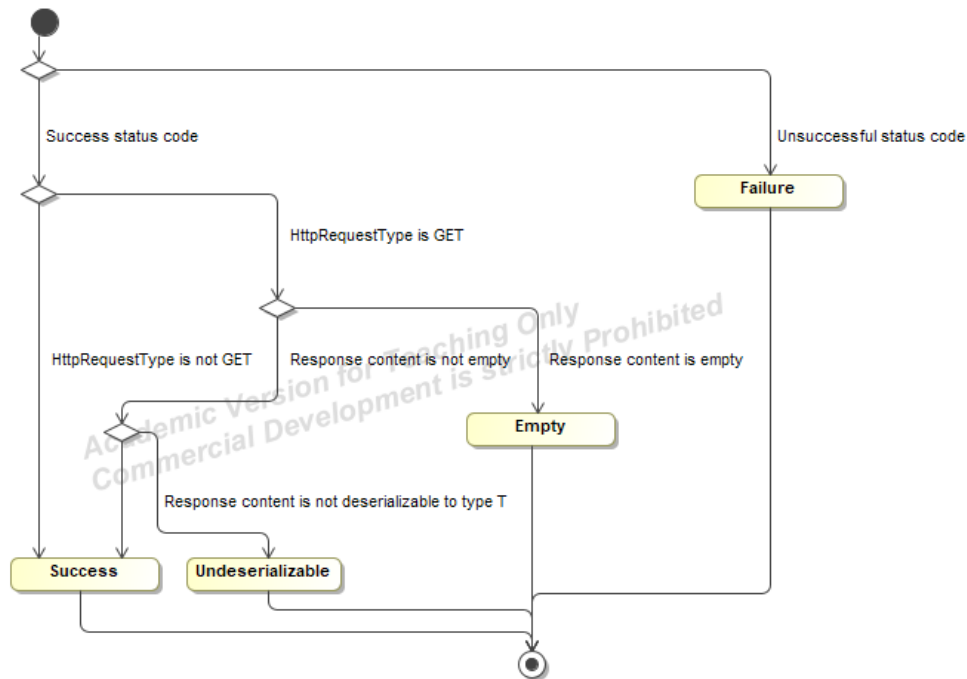
Komponento sekų diagrama pateikiama žemiau esančiame paveikslėlyje (2 pav.).



2 pav. Komponento sekų diagrama.

2.3. Būsenų diagrama

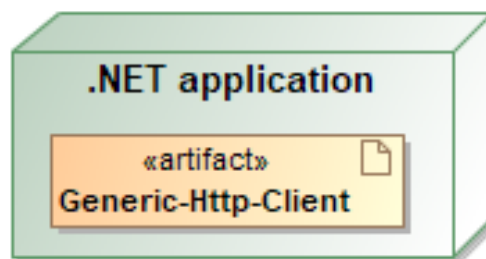
Komponento klasės HttpResponseMessage būsenų diagrama pateikiama žemiau esančiame paveikslėlyje (**3 pav.**).



3 pav. Komponento klasės HttpResponseMessage būsenų diagrama.

2.4. Išdėstymo diagrama

Komponento išdėstymo diagrama pateikiama žemiau esančiame paveikslėlyje (**4 pav.**).



4 pav. Komponento išdėstymo diagrama.

2.5. Kita papildoma informacija ir modeliai, kuriais remiantis galima būtų suprasti kaip komponentas funkcionuoja

Prieš inicializuojant komponento GenericHttpClient klasę, reikia inicializuoti IHttpApiClientWrapper interfeisą išpildančios klasės objektą ir jį paduoti į GenericHttpClient klasės konstruktorių. IHttpApiClientWrapper interfeisas gali būti įgyvendinamas pagal naudotojo pageidavimus, tačiau tinkant paprastam komponento funkcionalumui, naudotojas gali naudoti jau sukurtą HttpClientWrapper klasę, kuri išpildo IHttpApiClientWrapper interfeisą. Visas GenericHttpClient funkcionalumas pasiekiamas per vienintelį viešą metodą SendRequestAsync.

3. Komponento aprašymas

3.1. Klasių diagramos

Klasių diagrama matoma poskyryje „2.1. Klasių diagrama“ **1 pav.**

3.2. Detalūs metodų, klasių, paketų aprašai

Komponento klasių aprašai pateikiami žemiau esančioje lentelėje (**1 lentelė**).

1 lentelė. Komponento klasių aprašai

Klasės pavadinimas	Klasės aprašymas	Metodo pavadinimas	Metodo aprašymas
HttpClientWrapper	Klasė, apgaubianti System.Net.Http.HttpClient klasę.	GetAsync	Atlieka GET HTTP operaciją.
		PostAsync	Atlieka POST HTTP operaciją.
		PutAsync	Atlieka PUT HTTP operaciją.
		DeleteAsync	Atlieka DELETE HTTP operaciją.
GenericHttpClient	Bendros paskirties HTTP kliento klasė, atsakinga už HTTP operacijų validavimą ir kvietimą, bei operacijos rezultatų apdorojimą – JSON objektų deserializaciją, klaidos statusų ir pranešimų apdorojimą ir grąžinimą.	SendRequestAsync	Atlieka GET/POST/PUT/DELETE HTTP operacijas priklausomai nuo paduotų parametrų. Priima URL, kurį norima kviesti, HttpRequestType enumeratoriaus reikšmę, ir objektą, kuris gali būti perduodamas PUT/POST operacijose, kaip parametras. Metodas yra bendrinio tipo (generic), todėl yra skirtas atlikti pagrindines HTTP operacijas su bet koku objekto tipu.
HttpResponse	Klasė, grąžinama naudotojui atgal SendRequestAsync metu.		
HttpRequestType	HTTP operacijos, kurią norima atlikti, tipas. Galimos reikšmės – GET, PUT, POST ir DELETE.		
HttpResponseType	Komponento apdoroto atsako tipas, kuris priklauso nuo SendRequestAsync metu įvykusių veiksmų. Galimos reikšmės – Success, Failure, Empty, Undeserializable.		

3.3. Dokumentacija

Pagrindinių komponento klasių dokumentacija pateikiama žemiau esančiuose paveikslėliuose (5 – 9 pav.)

```
namespace GenericHttpClient.Clients
{
    1 reference
    public interface IGenericHttpClient
    {
        /// <summary>
        /// Sends a request of provided to type to the provided url with the provided payload (optional).
        /// </summary>
        /// <param name="url">URL that will receive the request.</param>
        /// <param name="httpRequestType">HTTP request type.</param>
        /// <param name="payload">Payload to be send with this request.</param>
        /// <returns>HttpResponse containing request status, response type, data and message (in case of failure).</returns>
        13 references
        public Task<HttpResponse<T>> SendRequestAsync<T>(
            string url,
            HttpRequestType httpRequestType,
            T? payload = default);
    }
}
```

5 pav. Klasės „GenericHttpClient“ dokumentacija.

```
4 references
public interface IHttpApiClientWrapper
{
    /// <summary>
    /// Method wrapping System.Net.Http.HttpClient.GetAsync().
    /// </summary>
    /// <param name="url">URL to be called.</param>
    /// <returns>HttpResponseMessage.</returns>
    6 references
    public Task<HttpResponseMessage> GetAsync(string url);

    /// <summary>
    /// Method wrapping System.Net.Http.HttpClient.PostAsync().
    /// </summary>
    /// <typeparam name="T">Generic type.</typeparam>
    /// <param name="url">URL to be called.</param>
    /// <param name="payload">Payload to be passed.</param>
    /// <returns>HttpResponseMessage.</returns>
    4 references
    public Task<HttpResponseMessage> PostAsync<T>(string url, T? payload);

    /// <summary>
    /// Method wrapping System.Net.Http.HttpClient.PutAsync().
    /// </summary>
    /// <typeparam name="T">Generic type.</typeparam>
    /// <param name="url">URL to be called.</param>
    /// <param name="payload">Payload to be passed.</param>
    /// <returns>HttpResponseMessage.</returns>
    4 references
    public Task<HttpResponseMessage> PutAsync<T>(string url, T? payload);

    /// <summary>
    /// Method wrapping System.Net.Http.HttpClient.DeleteAsync().
    /// </summary>
    /// <param name="url">URL to be called.</param>
    /// <returns>HttpResponseMessage.</returns>
    4 references
    public Task<HttpResponseMessage> DeleteAsync(string url);
}
```

6 pav. Klasės „HttpApiClientWrapper“ dokumentacija.

```

4 references
public class HttpResponseMessage
{
    /// <summary>
    /// A type of response to be returned back to the caller.
    /// </summary>
    13 references
    public HttpResponseMessageType ResponseType { get; set; }

    /// <summary>
    /// Message in case of error to be returned back to the caller.
    /// </summary>
    7 references
    public string Message { get; set; } = string.Empty;

    /// <summary>
    /// Deserialized data of the provided type T to be returned back to the caller.
    /// </summary>
    13 references
    public T? Data { get; set; }
}

```

7 pav. Klasės „HttpResponse“ dokumentacija.

```

14 references
public enum HttpResponseMessage
{
    /// <summary>
    /// Returned if HttpResponseMessage Status was Successful and if further result actions went fine.
    /// </summary>
    Success,
    /// <summary>
    /// Returned if HttpResponseMessage Status was not Successful or if any of the request validations failed.
    /// </summary>
    Failure,
    /// <summary>
    /// Returned for the HTTP GET operation, if HttpResponseMessage Status was Successful, but the Content was null, empty or whitespace.
    /// </summary>
    Empty,
    /// <summary>
    /// Returned for the HTTP GET operation, if HttpResponseMessage Status was Successful, the Content was not null, empty or whitespace,
    /// but couldn't be deserialized into the provided type T.
    /// </summary>
    Undeserializable
}

```

8 pav. Enumeratoriaus „HttpResponseType“ dokumentacija.

```

32 references
public enum HttpRequestType
{
    /// <summary>
    /// Value corresponding HTTP GET Operation
    /// </summary>
    GET,
    /// <summary>
    /// Value corresponding HTTP POST Operation
    /// </summary>
    POST,
    /// <summary>
    /// Value corresponding HTTP PUT Operation
    /// </summary>
    PUT,
    /// <summary>
    /// Value corresponding HTTP DELETE Operation
    /// </summary>
    DELETE
}

```

9 pav. Enumeratoriaus „HttpRequestType“ dokumentacija.

3.4. Naudojimo pavyzdys

Komponento naudojimo pavyzdys pateikiamas žemiau esančiame paveikslėlyje (**10 pav.**)

```
using GenericHttpClient.Clients;
using GenericHttpClient.Models;

namespace GenericHttpClientExample
{
    0 references
    public class Program
    {
        0 references
        static async Task Main(string[] args)
        {
            var httpClientWrapper = new HttpClientWrapper();
            var genericHttpClient = new GenericHttpClient.Clients.GenericHttpClient(httpClientWrapper);
            var url = "https://api.example.com/data";
            var response = await genericHttpClient.SendRequestAsync<TestObject>(url, HttpRequestType.GET);

            if (response.ResponseType == HttpResponseMessage.Success)
            {
                var data = response.Data;
                // Handle success scenario
            }
            else
            {
                // Handle failure scenario
                Console.WriteLine(response.Message);
            }
        }

        1 reference
        private class TestObject
        {
            0 references
            public int property1 { get; set; }

            0 references
            public string property2 { get; set; }
        }
    }
}
```

10 pav. Komponento naudojimo pavyzdys.

3.5. Komponento vartojimo vadovas

1. Sukuriamas klasės HttpClientWrapper tipo objektas (**11 pav.**)

```
var httpClientWrapper = new HttpClientWrapper();
```

11 pav. HttpClientWrapper objekto sukūrimas.

2. Sukuriamas klasės GenericHttpClient tipo objektas (**12 pav.**)

```
var genericHttpClient = new GenericHttpClient.Clients.GenericHttpClient(httpClientWrapper);
```

12 pav. GenericHttpClient objekto sukūrimas.

3. Siunčiamos užklausos su pasirinktomis HTTP operacijomis (**13 pav.**)

```
var url = "https://api.example.com/data";
var exampleObjectToBeInserted = new TestObject
{
    property1 = 1,
    property2 = "test"
};

var getResponse = await genericHttpClient.SendRequestAsync<TestObject>(url, HttpRequestType.GET);
var putResponse = await genericHttpClient.SendRequestAsync(url, HttpRequestType.PUT, exampleObjectToBeInserted);
var postResponse = await genericHttpClient.SendRequestAsync(url, HttpRequestType.POST, exampleObjectToBeInserted);
var deleteResponse = await genericHttpClient.SendRequestAsync<bool>(url, HttpRequestType.DELETE);
```

13 pav. GET, PUT, POST ir DELETE HTTP užklausų siuntimas naudojantis GenericHttpClient komponentu.

3.6. Komponento diegimo instrukcijos ir papildomi reikalavimai naudojimui

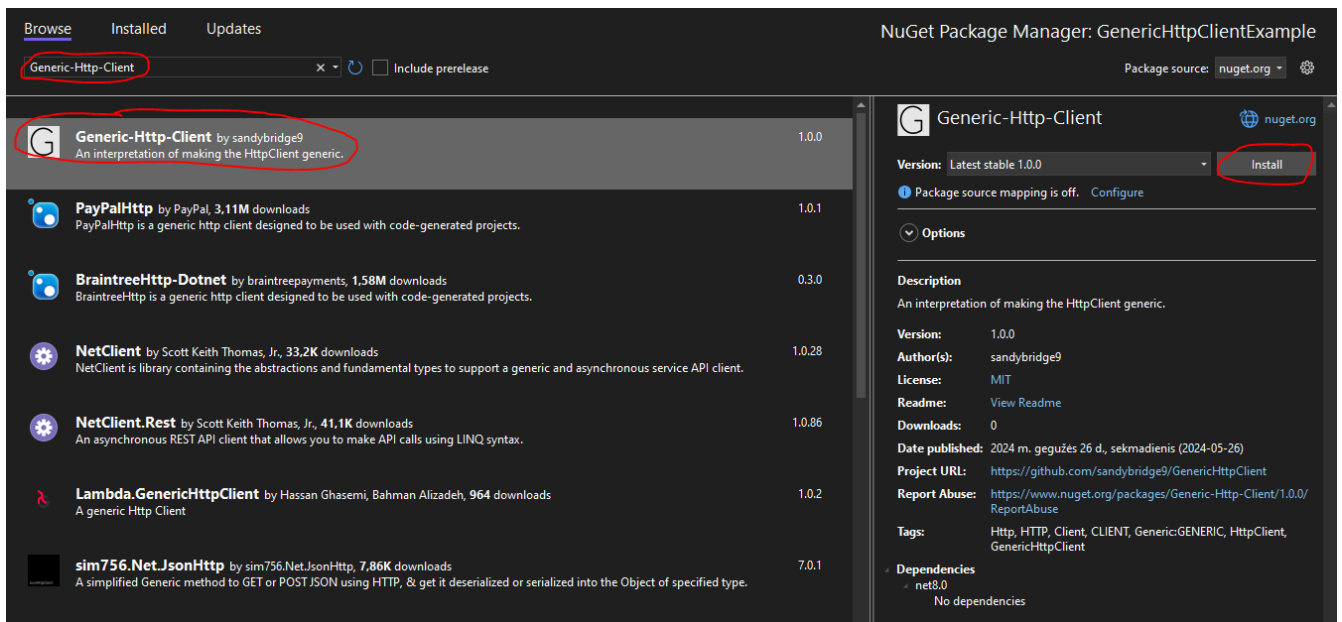
Prieš pradėdami naudoti komponentą, jį reikia įsidiegti. Išskiriami trys įdiegimo būdai: naudojantis .NET CLI (14 pav.), NuGet Package Manager (15 pav.) arba naudojantis Visual Studio NuGet Package Manager grafine sąsaja (16 pav.).

```
> dotnet add package Generic-Http-Client --version 1.0.0
```

14 pav. .NET CLI komponento diegimo būdas

```
PM> NuGet\Install-Package Generic-Http-Client -Version 1.0.0
```

15 pav. NuGet Package Manager diegimo būdas



















16 pav. Visual Studio NuGet Package Manager grafinės sąsajos diegimo būdas.

3.7. Reikalavimai diegimo aplinkai

Šis komponentas pritaikytas veikti .NET projektuose, turinčiuose bent .NET 8 versiją.

3.8. Pakeitimų istorija

Komponento Github pakeitimų istorija pateikiama žemiau pateiktame paveikslėlyje (**17 pav.**)

Activity	
 main	 All activity
 All users	 All time
Showing most recent first	
Updated GenericHttpClient to use static method + GenericHttpClient.cs...	
 sandybridge9 pushed 1 commit • d775fdb...bc0fe92 • 55 seconds ago	
Updated README.md with corrected package names and .csproj with READM...	
 sandybridge9 pushed 1 commit • 3fae2ac...d775fdb • 2 hours ago	
Update README.md	
 sandybridge9 pushed 1 commit • 06b34bb...3fae2ac • 2 hours ago	
Additional fixes for GenericHttpClient	
 sandybridge9 pushed 1 commit • fe4f16c...06b34bb • 2 hours ago	
Generic Http Client changes, added Wrapper + 100 percent unit test co...	
 sandybridge9 pushed 1 commit • 9eba102...fe4f16c • 10 hours ago	
Additional changes to the GenericHttpClient class and structure and p...	
 sandybridge9 pushed 1 commit • c0d8ca5...9eba102 • 23 hours ago	
Full implemented GenericHttpClient with GET POST PUT and DELETE reque...	
 sandybridge9 pushed 1 commit • 78002d5...c0d8ca5 • yesterday	
Extended HttpClientWrapper with additional functionality covering PUT...	
 sandybridge9 pushed 1 commit • ffad638...78002d5 • 2 days ago	
First version of GenericHttpClient	
 sandybridge9 pushed 1 commit • 52e79a5...ffad638 • 3 days ago	
Created a GenericHttpClient .NET solution and project	
 sandybridge9 pushed 1 commit • 611a052...52e79a5 • 3 days ago	
Create .gitignore	
 sandybridge9 pushed 1 commit • 22e9010...611a052 • 3 days ago	
Initial commit	
 sandybridge9 created this branch • 22e9010 • 3 days ago	

17 pav. Komponento Github pakeitimų istorija

3.9. Licencija

GenericHttpClient komponentui taikoma MIT licencija.

4. Rinkoje esamų komponentų apžvalga

.NET HttpClient klasę apgaubiančių ir jos funkcionalumą patobulinančių komponentų rinkoje yra daug, tačiau kiekvienas komponentas skirtingai praplečia HttpClient funkcionalumą, todėl skirtingose situacijose logiškiausia naudoti skirtingus komponentus. Peržvelgus vienus populiariausių HttpClient praplečiančių komponentų, randame pavyzdžių - pavyzdžiui Flurl.Http (<https://www.nuget.org/packages/Flurl.Http>) turi labai gerą URL konstravimo mechanizmą ir lengvą praplečiamumą, tačiau neturi įskiepytos JSON objektų deserializacijos ir paprasto klaidų valdymo mechanizmo. Tuo metu Polly.Extensions.Http (<https://www.nuget.org/packages/Polly.Extensions.Http>) turi labai gerą klaidų valdymo mechanizmą, tačiau neturi jokio URL konstravimo mechanizmo ir jokio bendrinio JSON objektų deserializavimo mechanizmo.

Didžioji dauguma .NET komponentų taip pat yra talpinami per nuget.org, kadangi tai yra pagrindinis komponentų talpinimo ir valdymo įrankis .NET platformai. Norint sukurti naują NuGet komponentą, reikia sekti šių instrukcijų:

1. Sukurti klasės bibliotekos tipo projektą.
2. Sukonfiguruoti paketo parametrus projekto savybių lange:
 - a. Pridėti paketo pavadinimą
 - b. Pridėti viešai matomą paketo pavadinimą
 - c. Pridėti paketo versijavimo taisykles
 - d. Pridėti nuorodas
 - e. Pridėti paketo identifikatorių
 - f. Pridėti README.md failą
 - g. Pridėti licenciją
3. Sugeneruoti projekto .nupkg failą
4. Susikurti nuget.org paskyrą ir įkelti šį failą į nuget.org
5. Palaukti patvirtinimo kad komponentas sėkmingai patalpintas nuget.org talpykloje.

Šio projekto metu kurtas komponentas yra viešai pasiekiamas per nuorodą: <https://www.nuget.org/packages/Generic-HttpClient/>. Komponento kodas taip pat yra viešai prieinamas adresu: <https://github.com/sandybridge9/GenericHttpClient>. Dabartinė komponento NuGet versija – 1.0.0