Online on-demand Project Support

Aus der Praxis: ASP.NET Core

Web API

Thorsten Kansy (tkansy@dotnetconsulting.eu)

Meine Person- Thorsten Kansy

Freier Consultant, Software Architekt,
Entwickler, Trainer & Fachautor

Microsoft SQL Server

.NET Core

ASP.NET Core

Azure Cosmos DB

DOTNETCONSULTING

Mein Service- Ihr Benefit

- Individuelle Inhouse Trainings
- (Online on-demand) Projektbegleitung
- Beratung
  - Problemanalyse und Lösungen
  - Technologieentscheidungen

DOTNETCONSULTING

Agenda

- Übersicht
  - Swagger, OpenAPI, Sicherheit,, …
- Persönlicher Teil
  - Swagger
  - Globaler Exception Handler
  - Self Hosting
  - Application Key
  - SignalR
  - Asynchrones
  - K6
  - Health Checks

Vielleicht ist ja etwas dabei…

DOTNET CONSULTING

Wie ASP.NET Core?

- Laut Microsoft
  - Cross-platform
  - High-performance,
  - Open source

- Basis von
  - Web.API
  - MVC
  - Blazor
  - …

Swagger

Swagger

- OpenAPI
- Swashbuckle Middleware
- swagger.json
  - Client generieren lassen
  - swagger-codegen

```
https://marketplace.visualstudio.com/items?itemName=ChristianResmaHelle.ApiClientCodeGenerator
```

Demo

Globaler Exception Handler

Globaler Exception Handler (GEH)

- Zentrale Verarbeitung von Ausnahmen
- Vereinheitliche Fehlerinfos
- Optimaler Weise auswertbar am Client

GEH: ConfigureServices

```
0 references | Thorsten, 251 days ago | 1 author, 2 changes
public void ConfigureServices(IServiceCollection services)
{
    // Set up controllers
    services.AddControllers(o =>
    {
        o.Filters.Add(new RestrictionFilter());
        o.Filters.Add(new ExceptionHandler());
    });
```

GEH: Exception Handler

```
public class ExceptionHandler : ExceptionFilterAttribute
{
    0 references | Thorsten, 251 days ago | 1 author, 2 changes
    public override Task OnExceptionAsync(ExceptionContext context)
    {
        if (context.Exception is CustomException exception)
        {
            context.Result = new JsonResult(ErrorDetails.Create(exception));
            context.HttpContext.Response.StatusCode = 400;
        }
        else if (context.Exception is ArgumentOutOfRangeException)
            context.Result = new NotFoundResult();

        return base.OnExceptionAsync(context);
    }
}
```

GEH: Error Details

```
public class ErrorDetails
{
    3 references | Thorsten, 252 days ago | 1 author, 1 change
    public int Id { get; set; }

    2 references | Thorsten, 252 days ago | 1 author, 1 change
    public string Message { get; set; }

    2 references | Thorsten, 252 days ago | 1 author, 1 change
    public object Details { get; set; }

    1 reference | Thorsten, 251 days ago | 1 author, 1 change
    public static ErrorDetails Create(ExternalApi.CustomException ex)
    {
        return new ErrorDetails
        {
            Id = ex.EventId.Id,
            Message = $"E: {ex.Message}",
            Details = ex.Details
        };
    }
```

GEH: Custom Exception

```
12 references | Thorsten, 251 days ago | 1 author, 1 change
public class CustomException : Exception
{
    public readonly EventId EventId;
    public readonly object? Details;

    4 references | Thorsten, 251 days ago | 1 author, 1 change
    public CustomException(EventId EventId, object? Details = null)
    {
        this.EventId = EventId;
        this.Details = Details!;
    }
}
```

Demo

Asynchrones

Asynchrones

```
[HttpPost(nameof(Delete))]
[Produces("application/json")]
1 reference | 0 changes | 0 authors, 0 changes
public async Task<IActionResult> Create(Specification Specification)
{
    _logger.LogInformation($"{nameof(Create)} ({Specification})");

    await _externalLogic.CreateSpecificationAsync(Specification);

    return Ok();
}
```

Aber was ist bei einem Abbruch?

Asynchrones mit Abbruch

```
public async Task<IActionResult> Create(CancellationToken cancellationToken)
{
    EntryDto trackingEntry = await _appLogic.CreateNewTrackingEntryAsync(cancellationToken);
    cancellationToken.ThrowIfCancellationRequested();

    IList<OrderDto> listOrder = await _appLogic.GetOrdersAsync(cancellationToken);
    cancellationToken.ThrowIfCancellationRequested();
```

```
// Ergebnis abrufen
IList<Order> rawResult = await query.ToListAsync(cancellationToken);
cancellationToken.ThrowIfCancellationRequested();
```

Http Status?

```
3 references | Thorsten, 29 days ago | 1 author, 1 change
public class OperationCancelledExceptionFilter : ExceptionFilterAttribute
{
    private readonly ILogger _logger;

    0 references | Thorsten, 29 days ago | 1 author, 1 change
    public OperationCancelledExceptionFilter(ILoggerFactory loggerFactory)
    {
        _logger = loggerFactory.CreateLogger<OperationCancelledExceptionFilter>();
    }

    0 references | Thorsten, 29 days ago | 1 author, 1 change
    public override void OnException(ExceptionContext context)
    {
        if (context.Exception is OperationCanceledException)
        {
            _logger.LogInformation("Request was cancelled");
            context.ExceptionHandled = true;
            context.Result = new StatusCodeResult(499); // 499 CLIENT CLOSED REQUEST (non standard)
        }
    }
}
```

Demo

Self hosting

Self hosting

- Mehrere (unabhängige) Web.APIs in einem Programm
  - Service/ Demon

Self hosting: AddHostedService

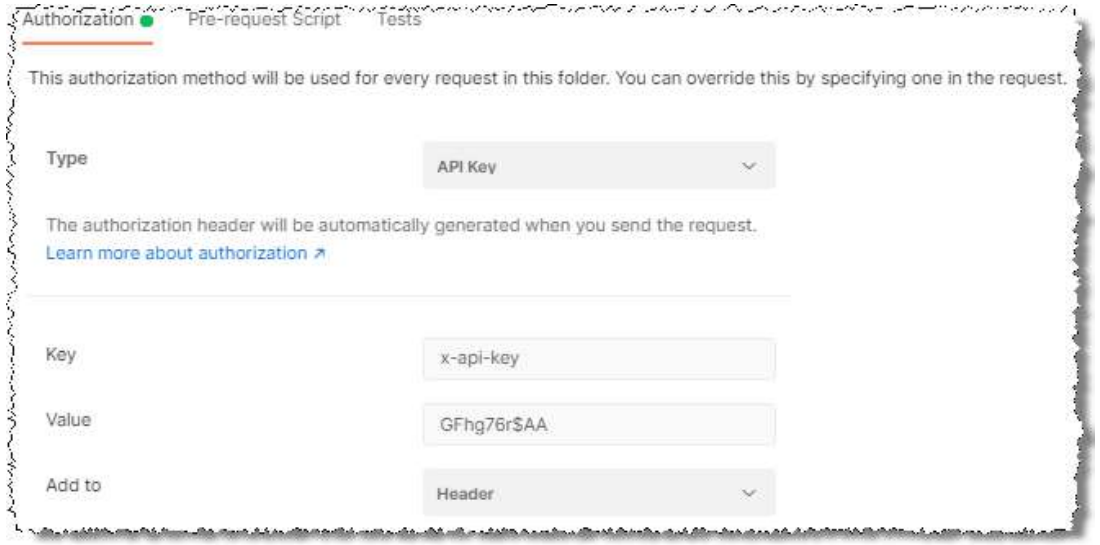
```
1 reference | 0 changes | 0 authors, 0 changes
public static IHostBuilder CreateHostBuilder(string[] args)
{
    return Host.CreateDefaultBuilder(args)
        .ConfigureServices((hostContext, services) =>
        {
            // External Api
            IConfigurationSection externalApiConfigurationSection = Configuration.GetSection("ExternalApi");
            services.Configure<ExternalApi.WebApiSettings>(externalApiConfigurationSection);
            services.AddHostedService<ExternalApi.WebApiService>();

            // Internal Api
            IConfigurationSection internalApiConfigurationSection = Configuration.GetSection("InternalApi");
            services.Configure<InternalApi.WebApiSettings>(internalApiConfigurationSection);
            services.AddHostedService<InternalApi.WebApiService>();
        });
}
```

Demo

Application Key

Application Key

- Sicherstellen/ Kontrollieren welche Anwendung welche API verwendet
- `x-api-key` Header
- Schwache Sicherheitsfeatures



Application Key: IAuthorizationFilter

```
/// <summary>
/// Filter to check api key.
/// </summary>
5 references | 0 changes | 0 authors, 0 changes
public class ApiKeyFilter : IAuthorizationFilter
{
    /// <summary>
    /// Name of header.
    /// </summary>
    public const string APIKEYNAME = "x-api-key";
```

Application Key: OnAuthorization

```
0 references | 0 changes | 0 authors, 0 changes
public void OnAuthorization(AuthorizationFilterContext context)
{
    bool skipApiKeyCheck = false;

    // Currently we only skip the check if the Info-Controller is invoked.
    if (context.ActionDescriptor is ControllerActionDescriptor cad)
    {
        // Skip for InfoController
        skipApiKeyCheck = cad.ControllerTypeInfo.AsType() == typeof(Common.InfoController);
    }

    if (skipApiKeyCheck)
        return;

    // Verify API key
    string apiKey = context.HttpContext.Request.Headers[APIKEYNAME].ToString();

    if (string.Compare(_apiKey, apiKey) != 0)
        context.Result = new UnauthorizedResult();
}
```

Application Key: ConfigureServices

```
0 references | Thorsten, 251 days ago | 1 author, 2 changes
public void ConfigureServices(IServiceCollection services)
{
    // Set up controllers
    services.AddControllers(o =>
    {
        o.Filters.Add(new RestrictionFilter());
        o.Filters.Add(new ExceptionHandler());
    });
```

IExceptionHandler-Middleware

8.0

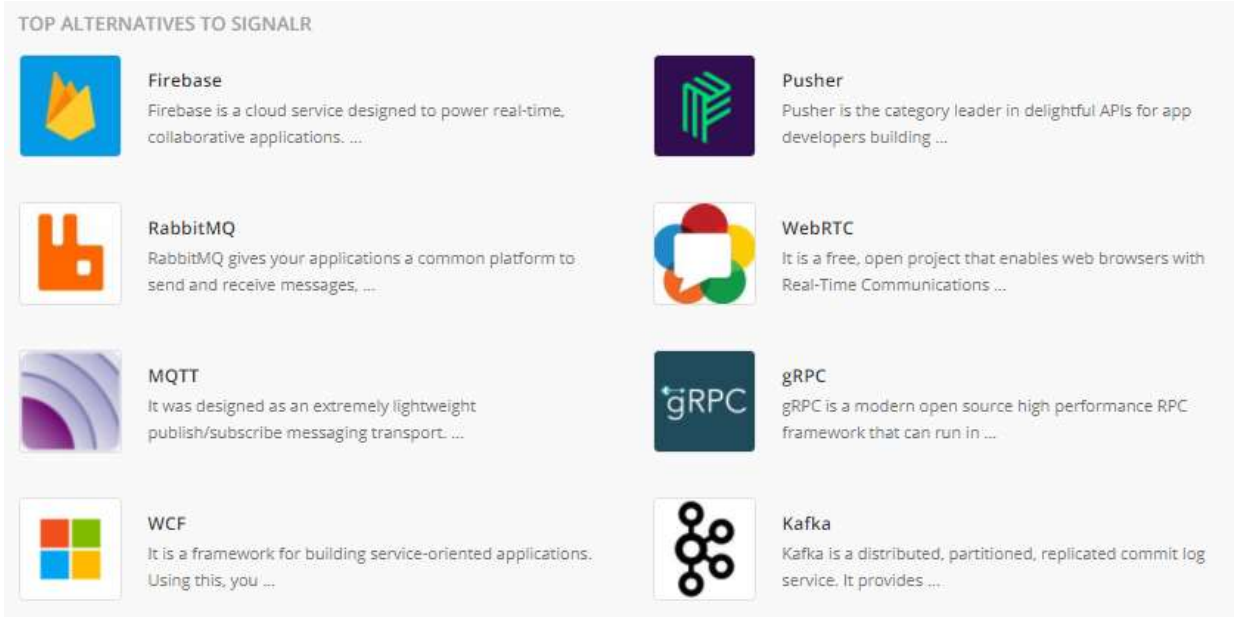
```
public class GlobalExceptionHandler(ILogger<GlobalExceptionHandler> logger) : IExceptionHandler
{
    0 references | 0 changes | 0 authors, 0 changes
    public async ValueTask<bool> TryHandleAsync(
        HttpContext httpContext,
        Exception exception,
        CancellationToken cancellationToken)
    {
        logger.LogError(exception, "Exception occurred: {Message}", exception.Message);

        var problemDetails = new ProblemDetails
        {
            Status = StatusCodes.Status500InternalServerError,
            Title = "Server error"
        };

        httpContext.Response.StatusCode = problemDetails.Status.Value;

        await httpContext.Response
            .WriteAsJsonAsync(problemDetails, cancellationToken);

        return true;
    }
```

Demo

SignalR

SignalR

- Echtzeitverbindung Web.API => Client
- Nachrichten gehen vom Server (HUB) aus



SignalR: Server (Hub)

```
[Authorize(Policy = "ApiKeyPolicy")]
7 references | 0 changes | 0 authors, 0 changes
public class SignalRHub : Hub<ISignalRMethods>
{
    // Endpoint to register to
    public const string ENDPOINT = "/ExternalHub";
    private readonly ILogger<SignalRHub> _logger;

    0 references | 0 changes | 0 authors, 0 changes
    public SignalRHub(ILogger<SignalRHub> logger)
    {
        _logger = logger;
        _logger.LogInformation("SignalR hub started.");
    }
}
```

```
5 references | 0 changes | 0 authors, 0 changes
public interface ISignalRMethods
{
    1 reference | 0 changes | 0 authors, 0 changes
    Task NewScan();
}
```

```
public static IHubContext<SignalRHub, ISignalRMethods> externalSignalRHub = null!;
```

```
await Common.GlobalObjects.externalSignalRHub.Clients.All.NewScan();
```

SignalR: Client

```
#region Setup SignalR Connection
connection = new HubConnectionBuilder()
    .WithUrl(SIGNALRHUB, options =>
    {
        options.Headers.Add("x-api-key", apiKey);
    })
    .WithAutomaticReconnect(new TimeSpan[] { TimeSpan.FromSeconds(10) })
    .Build();

connection.Closed += async (error) =>
{
    await Task.Delay(new Random().Next(0, 5) * 1000);
    await connection.StartAsync();
};

connection.Reconnecting += error =>
{
    Console.WriteLine($"Error: connection.State == {connection.State}");
    return Task.CompletedTask;
};

connection.StartAsync().ContinueWith(t =>...).Wait();
#endregion
```

Demo

HashId

HashId

Vorhersagbare IDs vermeiden

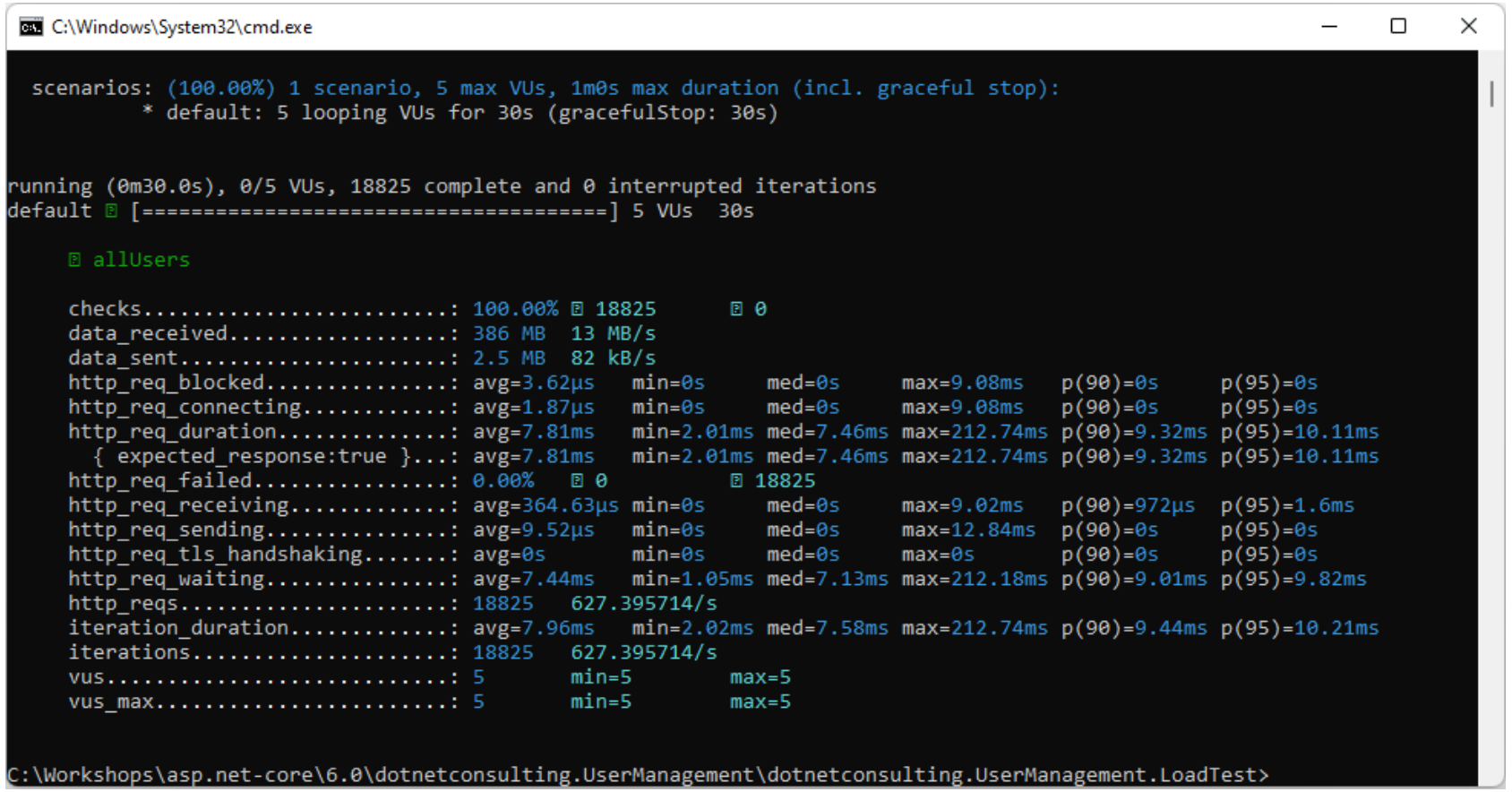
```
 ĤţţřĜêţ
 Řộuţê    îđ ḥắşḥîđş
řučľîç AçţîộŋRêşuľţ CuşţộņêsDţộ
        Ĝêţ  ŇộđêľBîŋđês ţỳřêộğ ĤắşḥîđşŇộđêľBîŋđês     îŋţ îđ


     sêţuşŋ Ôľ çuşţộņêşş ŞîŋĝľêÔsDêğắuľţ ç     ç Íđ     îđ
```

```
Install-Package AspNetCore.Hashids
```

Demo

K6 Load test

K6 Load test



https://k6.io

Demo

Health Checks

Health Checks

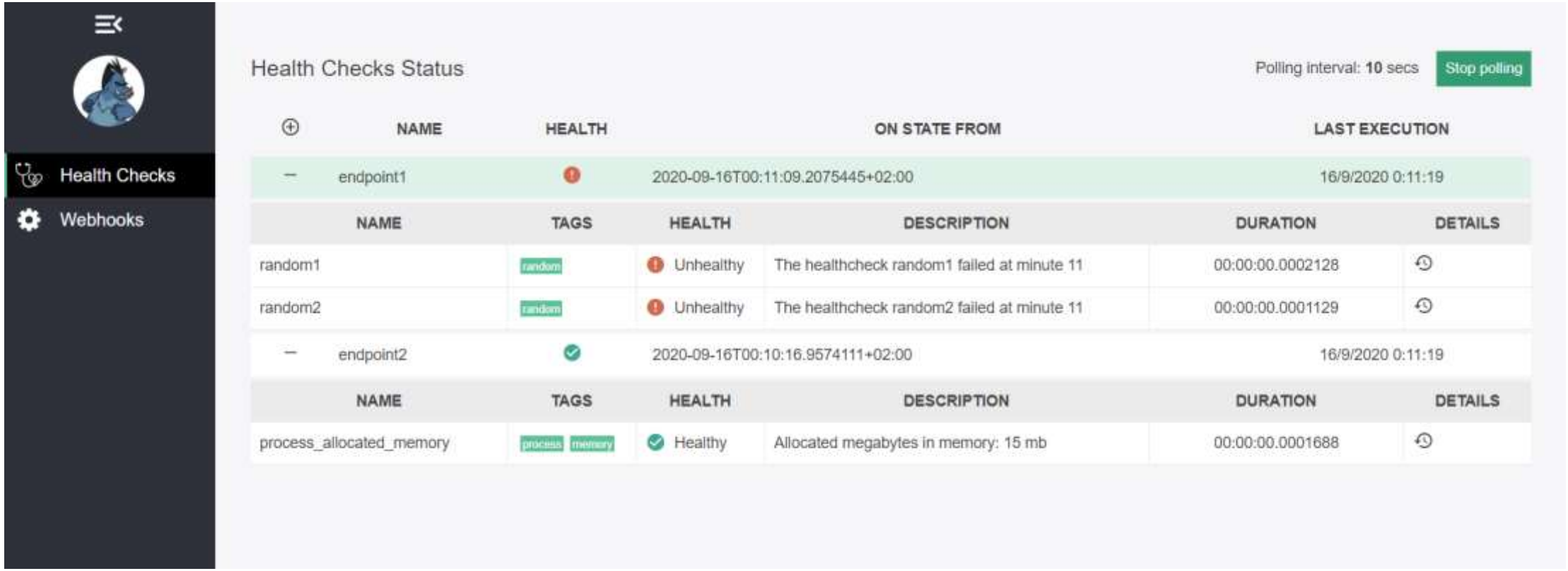
Prüfen, ob die genutzten Ressourcen verfügbar sind

```
čụîľđês Şêsŵîçês AđđĤêǎľţḫChêçlș
    Ačês xîê  șộľľ xêľçḥês Ţêșţ đụsçḥĝêğứḥsţ xêsđêŋ
    Íŋřľêŋêŋţĵîêsụŋĝ ŵộŋ ÍĤêǎľţḫChêçl

ǎřř ŇǎřĤêǎľţḫChêçlș    ḥêǎľţḫ
```

```
Install-Package AspNetCore.HealthChecks.*
```

https://github.com/Xabaril/AspNetCore.Diagnostics.HealthChecks

UI




```
Install-Package AspNetCore.HealthChecks.UI.*
```

Fragen? Jetzt oder später!

Kontakt

E-Mail
tkansy@dotnetconsulting.eu

Telefon
+49 (0) 6187 / 2009090

Microsoft Teams
Meet now

LinkedIn
Link me

XING
Xing me

X (Twitter)
@tkansy

DOTNETCONSULTING

www.dotnetconsulting.eu

SQL Server meets .NET (Core)- professionally!

Ich berate, coache und trainiere im Bereich Entwicklung von .NET (Core) Anwendungen mit Microsoft SQL Server- mit Allem, was dazu gehört- und was man vielleicht weglassen sollte.

DOTNETCONSULTING