

What is new in .NET Core 3(.1) and the future of .NET

Johnny Hooyberghs

involved

tech·days

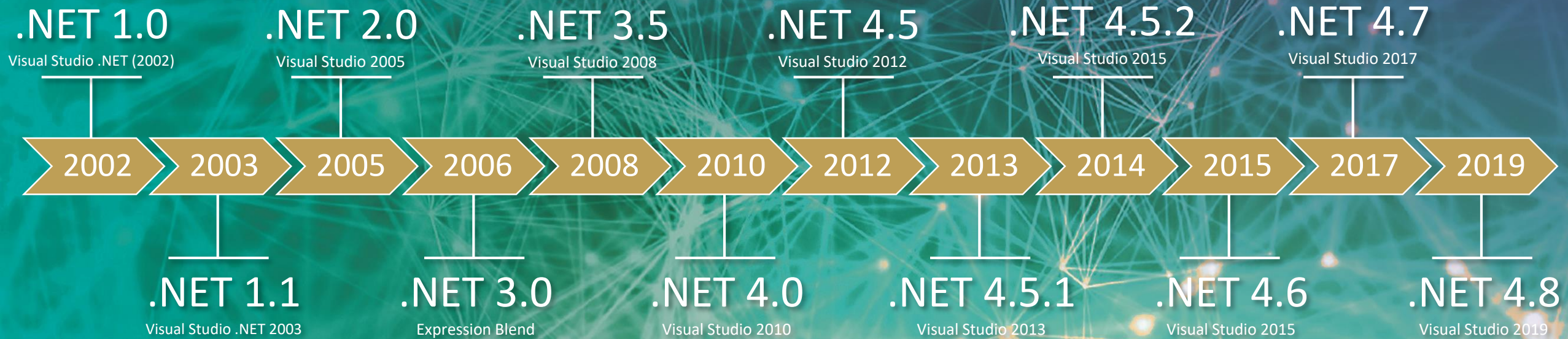
Powered by Microsoft

www.involved-it.be

Agenda

- .NET: a quick history
- .NET: what about the future?
- .NET Core 3.0 & 3.1
 - New publish features
 - Ranges and Indices
 - Async streams (iterators)
 - Platform-Dependent Intrinsic
 - Built-in JSON support
 - HTTP/2 and gRPC
 - Worker Services
 - Windows Desktop

.NET: a quick history



.NET: a quick history

.NET Core 1.0

Visual Studio 2015 Update 3

.NET Core 1.1

Visual Studio 2017 Version 15.0

.NET Core 2.0

Visual Studio 2017 Version 15.3

.NET Core 2.1

Visual Studio 2017 Version 15.7

.NET Core 2.2

Visual Studio 2019 Version 16.0

.NET Core 3.0

Visual Studio 2019 Version 16.3

.NET Core 3.1

Visual Studio 2019 Version 16.4

2016

2016

2017

2018

2018

2019

2019

.NET Core 1.0

June 27, 2019

.NET Core 1.1

June 27, 2019

.NET Core 2.0

October 1, 2018

.NET Core 2.1

August 21, 2021

.NET Core 2.2

December 23, 2019

.NET Core 3.0

March 3, 2020

.NET Core 3.1

Supported

tech·days

Powered by Microsoft

involved

.NET: what about the future?



tech·days

Powered by Microsoft

THIS DEMO USES

**"PRODUCTION-READY
CODE"**

New Publish Features: Default executables

- Since .NET Core 3.0
- Build and Publish
- DLL that can be executed using “dotnet”
 - platform independent
- Platform SDK decides how to build native executable
 - Platform dependent

involved

New Publish Features: Default executables

.NET Core 3.1

Name	Date modified	Type	Size
01-DefaultExecutables.Common.dll	17/12/2019 12:37	Application extens...	5 KB
01-DefaultExecutables.Common.pdb	17/12/2019 12:37	Program Debug D...	1 KB
01-DefaultExecutables.DotNetCoreThree.deps.json	17/12/2019 12:37	JSON File	1 KB
01-DefaultExecutables.DotNetCoreThree.dll	17/12/2019 12:37	Application extens...	5 KB
01-DefaultExecutables.DotNetCoreThree.exe	17/12/2019 12:37	Application	67 KB
01-DefaultExecutables.DotNetCoreThree.pdb	17/12/2019 12:37	Program Debug D...	1 KB
01-DefaultExecutables.DotNetCoreThree.runtimeconfig.dev.json	17/12/2019 12:37	JSON File	1 KB
01-DefaultExecutables.DotNetCoreThree.runtimeconfig.json	17/12/2019 12:37	JSON File	1 KB

.NET Core 2.2

Name	Date modified	Type	Size
01-DefaultExecutables.Common.dll	17/12/2019 12:37	Application extens...	5 KB
01-DefaultExecutables.Common.pdb	17/12/2019 12:37	Program Debug D...	1 KB
01-DefaultExecutables.DotNetCoreTwo.deps.json	17/12/2019 12:37	JSON File	1 KB
01-DefaultExecutables.DotNetCoreTwo.dll	17/12/2019 12:37	Application extens...	5 KB
01-DefaultExecutables.DotNetCoreTwo.pdb	17/12/2019 12:37	Program Debug D...	1 KB
01-DefaultExecutables.DotNetCoreTwo.runtimeconfig.dev.json	17/12/2019 12:37	JSON File	1 KB
01-DefaultExecutables.DotNetCoreTwo.runtimeconfig.json	17/12/2019 12:37	JSON File	1 KB

New Publish Features: Single File Executables

- Since .NET Core 3.0
- Publish
- Self-extracting executable containing all dependencies
- Extracts to temp location on disk
- Includes .NET Core runtime
- Platform SDK decides how to build native executable

involved



New Publish Features: Single File Executables

```
<PropertyGroup>
  <OutputType>Exe</OutputType>
  <TargetFramework>netcoreapp3.1</TargetFramework>
  <RootNamespace> 02_SingleFileExecutables</RootNamespace>
  <RuntimeIdentifier>win10-x64</RuntimeIdentifier>
  <PublishSingleFile>true</PublishSingleFile>
</PropertyGroup>
```

OR

```
dotnet publish -r win10-x64 -p:PublishSingleFile=true
```



Name	Date modified	Type	Size
 02-SingleFileExecutables.exe	16/12/2019 11:33	Application	67.476 KB
 02-SingleFileExecutables.pdb	16/12/2019 11:33	Program Debug D...	1 KB

Platform-Dependent Ininsics

- Since .NET Core 1.0
 - SIMD (Single Instruction Multiple Data)
 - System.Numerics
 - Fallback if not supported on CPU
- Since .NET Core 3.0
 - More extensive support for SIMD
 - System.Runtime.Intrinsics
 - No fallback if not supported on CPU

Platform-Dependent Intrinsic

Getting a billion integers...

Calculating a regular sum... 4941ms

Calculating a sum with SIMD support... 2777ms

Calculating a sum with Hardware Intrinsic support... 1573ms

Calculating a regular sum... 4911ms

Calculating a sum with SIMD support... 2753ms

Calculating a sum with Hardware Intrinsic support... 1529ms

Calculating a regular sum... 4743ms

Calculating a sum with SIMD support... 2718ms

Calculating a sum with Hardware Intrinsic support... 1550ms

... days

Built-in JSON support

- Since .NET Core 3.0
- Applied by default in ASP.NET Core 3.0 for serialization
- No more dependency on JSON.NET (Newtonsoft)
- System.Text.Json
- Doesn't rely on System.String (UTF-16)
- Uses Span<byte> to manipulate strings (UTF-8)
- Can still be replaced by JSON.NET if wanted

Built-in JSON support

Preparing data...

Newtonsoft JSON serialization:	563ms
Newtonsoft JSON deserialization:	206ms
Built-in JSON serialization:	156ms
Built-in JSON deserialization:	159ms

Newtonsoft JSON serialization:	160ms
Newtonsoft JSON deserialization:	159ms
Built-in JSON serialization:	102ms
Built-in JSON deserialization:	89ms

Newtonsoft JSON serialization:	106ms
Newtonsoft JSON deserialization:	126ms
Built-in JSON serialization:	97ms
Built-in JSON deserialization:	83ms

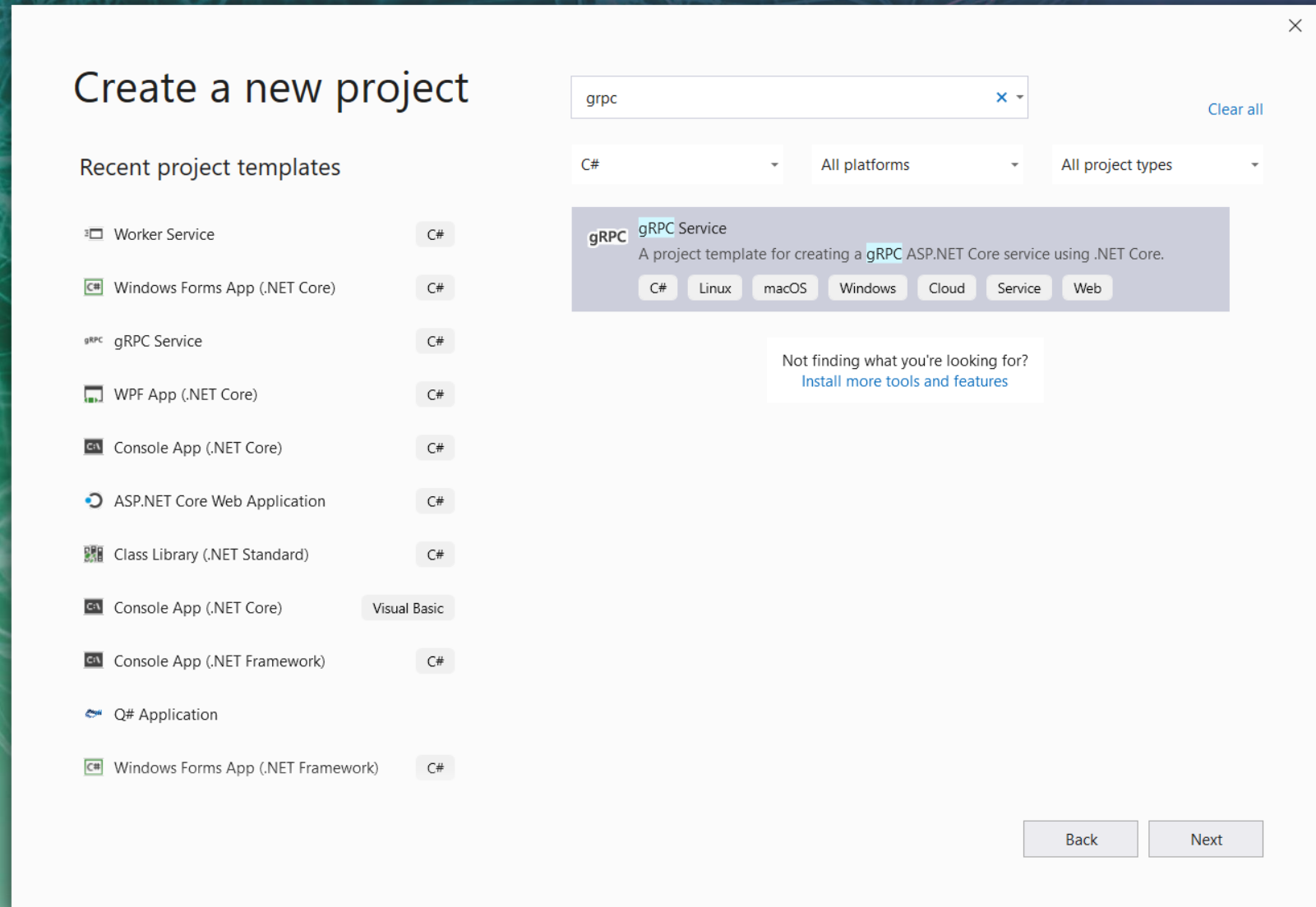
tech·days

Powered by Microsoft

HTTP/2 and gRPC

- Since ASP.NET Core 3.0
- OpenSource Remote Procedure Call system by Google
- Protocol buffers as Interface Description Language
- Needs HTTP/2
- Needs Transport Layer Security (TLS)
- Migration path for .NET Framework WCF to .NET Core gRPC
- Supported by Kestrel
- Currently not supported on Azure App Service and IIS
 - Preview/experimental support for gRPC-Web since January 27th, 2020

HTTP/2 and gRPC



HTTP/2 and gRPC

```
syntax = "proto3";

option csharp_namespace = "_10_gRPC.Proto";

package contract;

service Sauna {
  rpc FetchCurrentState (SaunaRequest) returns (SaunaResponse);
  rpc FetchStateStream (SaunaRequest) returns (stream SaunaResponse);
}

message SaunaRequest {
  string temperatureUnit = 1;
}

message SaunaResponse {
  int64 timeStamp = 1;
  bool isDrySauna = 2;
  bool isInfraRed = 3;
  int32 temperature = 4;
  string description = 5;
}
```


HTTP/2 and gRPC

```
<ItemGroup>  
  <Protobuf Include="Protos\contract.proto" GrpcServices="Server" />  
</ItemGroup>
```

```
<ItemGroup>  
  <PackageReference Include="Google.Protobuf" Version="3.11.2" />  
  <PackageReference Include="Grpc.AspNetCore" Version="2.25.0" />  
</ItemGroup>
```

```
<ItemGroup>  
  <PackageReference Include="Google.Protobuf" Version="3.11.2" />  
  <PackageReference Include="Grpc.Net.Client" Version="2.25.0" />  
  <PackageReference Include="Grpc.Tools" Version="2.25.0">  
    <PrivateAssets>all</PrivateAssets>  
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>  
  </PackageReference>  
</ItemGroup>
```

```
<ItemGroup>  
  <Protobuf Include="..\10-gRPC.Server\Protos\contract.proto" GrpcServices="Client">  
    <Link>Protos\contract.proto</Link>  
  </Protobuf>  
</ItemGroup>
```


HTTP/2 and gRPC

```
app.UseRouting();

app.UseEndpoints(endpoints =>
{
    endpoints.MapGrpcService<SaunaService>();

    endpoints.MapGet(pattern: "/", requestDelegate: async context =>
    {
        await context.Response.WriteAsync(text: "Communication with gRPC endpoints");
    });
});
```

```
public override Task<SaunaResponse> FetchCurrentState(SaunaRequest request, ServerCallContext context)
{
    return Task.FromResult(new SaunaResponse
    {
        TimeStamp = DateTimeOffset.UtcNow.ToUnixTimeSeconds(),
        IsDrySauna = _randomGenerator.Next(0, 2) == 1,
        IsInfraRed = _randomGenerator.Next(0, 2) == 1,
        Temperature = GetTemperature(request.TemperatureUnit),
        Description = ""
    });
}
```


HTTP/2 and gRPC

```
var channel = GrpcChannel.ForAddress("https://localhost:5001");  
var client = new Sauna.SaunaClient(channel);  
var request = new SaunaRequest { TemperatureUnit = "C" };  
var response = await client.FetchCurrentStateAsync(request);  
  
var timeStamp = DateTimeOffset.FromUnixTimeSeconds(response.TimeStamp);  
  
WriteLine($"[{timeStamp:T}] <{(response.IsInfraRed ? "I" : " ")}> " +  
    $"<{(response.IsDrySauna ? "S" : " ")}> {response.Temperature}°C");
```













Worker Services

- Since .NET Core 2.1
- IHostBuilder support for long running IHostedService
- No more need for .NET Core Console applications with `while(true)`
- Support for (Docker) containers
- Support for Windows Service deployments
- Support for Linux Daemon using `systemd`

Worker Services

Create a new project


Recent project templates


 Worker Service	C#
 Windows Forms App (.NET Core)	C#
 gRPC Service	C#
 WPF App (.NET Core)	C#
 Console App (.NET Core)	C#
 ASP.NET Core Web Application	C#
 Class Library (.NET Standard)	C#
 Console App (.NET Core)	Visual Basic
 Console App (.NET Framework)	C#
 Q# Application	
 Windows Forms App (.NET Framework)	C#


worker service


Clear all


C#All platformsAll project types


 **Worker Service**
A project template for creating a worker service using .NET Core.
C#LinuxmacOSWindowsCloudService

 **gRPC Service**
A project template for creating a gRPC ASP.NET Core service using .NET Core.
C#LinuxmacOSWindowsCloudServiceWeb

 **Service Fabric Application**
A project template for creating an always-on, scalable, distributed application with Microsoft Azure Service Fabric.
C#AzureCloud

 **Azure Cloud Service (classic)**
A project for creating a scalable service that runs on Microsoft Azure.
C#AzureCloud

 **Windows Service (.NET Framework)**
A project for creating Windows Services
C#WindowsDesktopService

 **WCF Service**

Next

Worker Services

0 references | Johnny Hooyberghs, 1 hour ago | 1 author, 1 change

```
public static void Main(string[] args)
{
    CreateHostBuilder(args).Build().Run();
}
```

1 reference | Johnny Hooyberghs, 1 hour ago | 1 author, 1 change

```
public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureServices((hostContext, services) =>
        {
            services.AddHostedService<WorkerAlpha>();
            services.AddHostedService<WorkerBeta>();
        });
```

4 references | Johnny Hooyberghs, 1 hour ago | 1 author, 1 change

```
protected override async Task ExecuteAsync(CancellationToken stoppingToken)
{
    while (!stoppingToken.IsCancellationRequested && _workCount < 10)
    {
        _logger.LogInformation(message: "Worker running at: {time}", DateTimeOffset.Now);
        await Task.Delay(1000, stoppingToken);
        _workCount++;
    }
}
```


Windows Desktop

- Since .NET Core 3.0
- Support for WinForms, WPF and UWP
- Only on Windows
 - Wrapper on top of Windows DirectX and GDI+ apis
- XAML designer included in Visual Studio 2019
- WinForms designer only included in Visual Studio 2019 as a preview extension (VSIX)

Windows Desktop

Create a new project

winforms

Clear all

Recent project templates

- Worker Service C#
- Windows Forms App (.NET Core) C#
- gRPC Service C#
- WPF App (.NET Core) C#
- Console App (.NET Core) C#
- ASP.NET Core Web Application C#
- Class Library (.NET Standard) C#
- Console App (.NET Core) Visual Basic
- Console App (.NET Framework) C#
- Q# Application
- Windows Forms App (.NET Framework) C#

C#

All platforms

All project types



Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

C# Windows Desktop



Windows Forms App (.NET Core)

A project for creating an application with a Windows Forms (WinForms) user interface

C# Windows Desktop



Windows Forms Control Library (.NET Framework)

A project for creating controls to use in Windows Forms (WinForms) applications

C# Windows Desktop Library



Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

Visual Basic Windows Desktop



Windows Forms Control Library (.NET Framework)

A project for creating controls to use in Windows Forms (WinForms) applications

Visual Basic Windows Desktop Library

Next

Create a new project

wpf

Clear all

Recent project templates

- Worker Service C#
- Windows Forms App (.NET Core) C#
- gRPC Service C#
- WPF App (.NET Core) C#
- Console App (.NET Core) C#
- ASP.NET Core Web Application C#
- Class Library (.NET Standard) C#
- Console App (.NET Core) Visual Basic
- Console App (.NET Framework) C#
- Q# Application
- Windows Forms App (.NET Framework) C#

C#

All platforms

All project types



WPF User Control Library (.NET Core)

Windows Presentation Foundation user control library

C# Windows Desktop Library



WPF Custom Control Library (.NET Core)

Windows Presentation Foundation custom control library

C# Windows Desktop Library



WPF App (.NET Core)

Windows Presentation Foundation client application

C# Windows Desktop



WPF App (.NET Framework)

Windows Presentation Foundation client application

C# Windows Desktop



WPF Browser App (.NET Framework)

Windows Presentation Foundation browser application

C# Windows Desktop



WPF Custom Control Library (.NET Framework)

Windows Presentation Foundation custom control library

Next

johnny.hooyberghs@involved-it.be
@djohnnieke

<https://github.com/Djohnnie>

<https://github.com/Djohnnie/DotNetCore3-TechDaysFinland-2020>



involved

tech·days

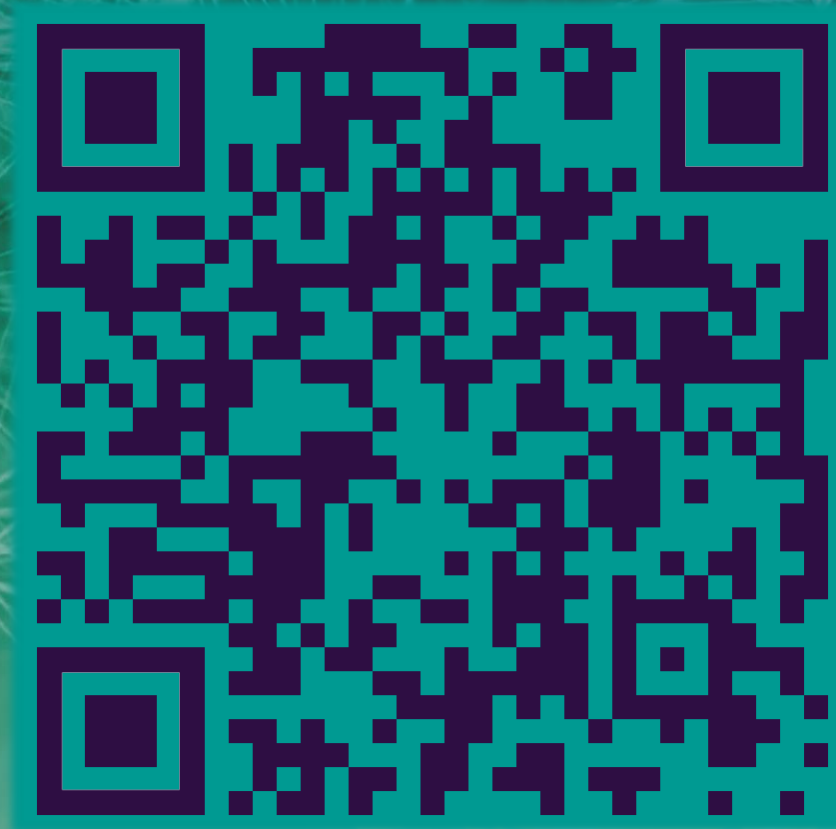
Powered by Microsoft

www.involved-it.be

johnny.hooyberghs@involved-it.be
@djohnnieke

<https://github.com/Djohnnie>

<https://github.com/Djohnnie/DotNetCore3-TechDaysFinland-2020>



involved

tech·days

Powered by Microsoft

www.involved-it.be