

Adding a UI



Gill Cleeren

CTO Xebia Microsoft Services Belgium

@gillcleeren

Overview



Introducing Blazor WebAssembly

Using NSwag and NSwagStudio

Exploring the client app

Adding a missing feature





Introducing Blazor WebAssembly



**Blazor is a framework
to build full stack web apps
using C# and HTML and
without JavaScript.**

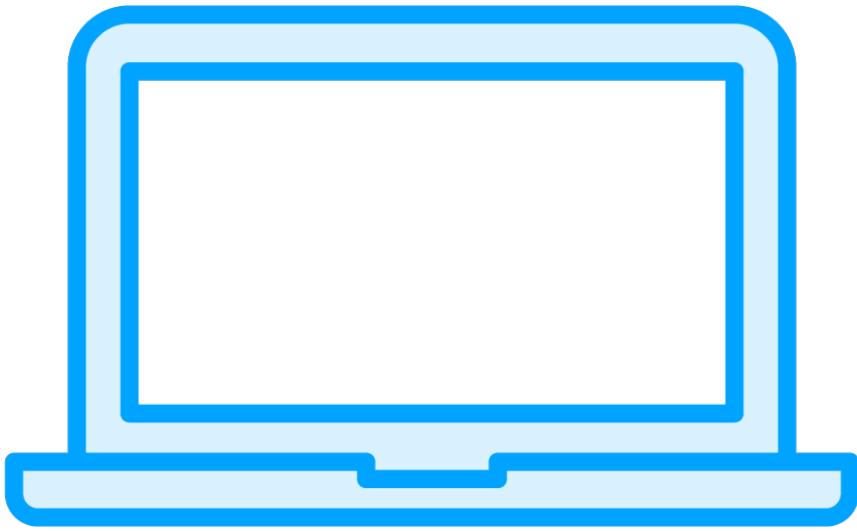


Blazor Hosting Models

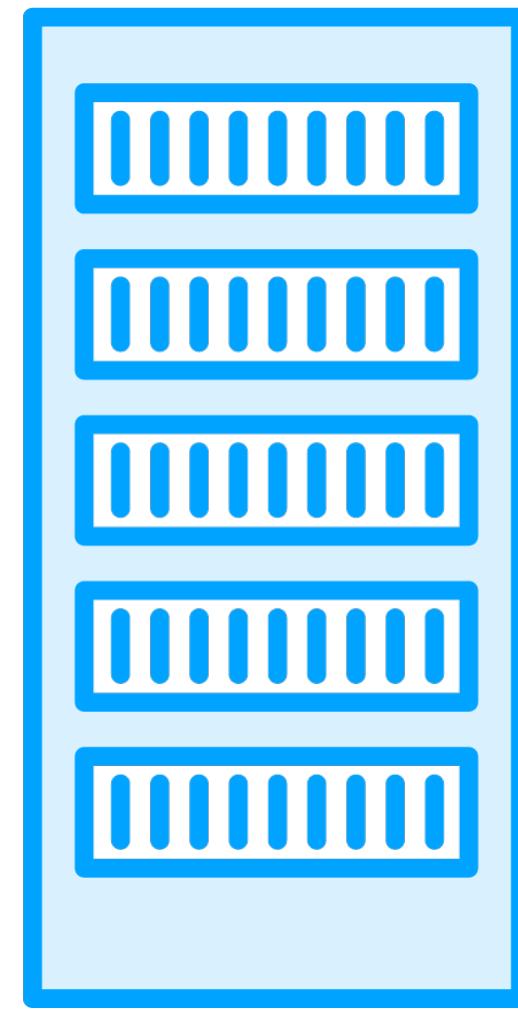
Blazor Client
WASM



Blazor WebAssembly



Blazor app
WebAssembly



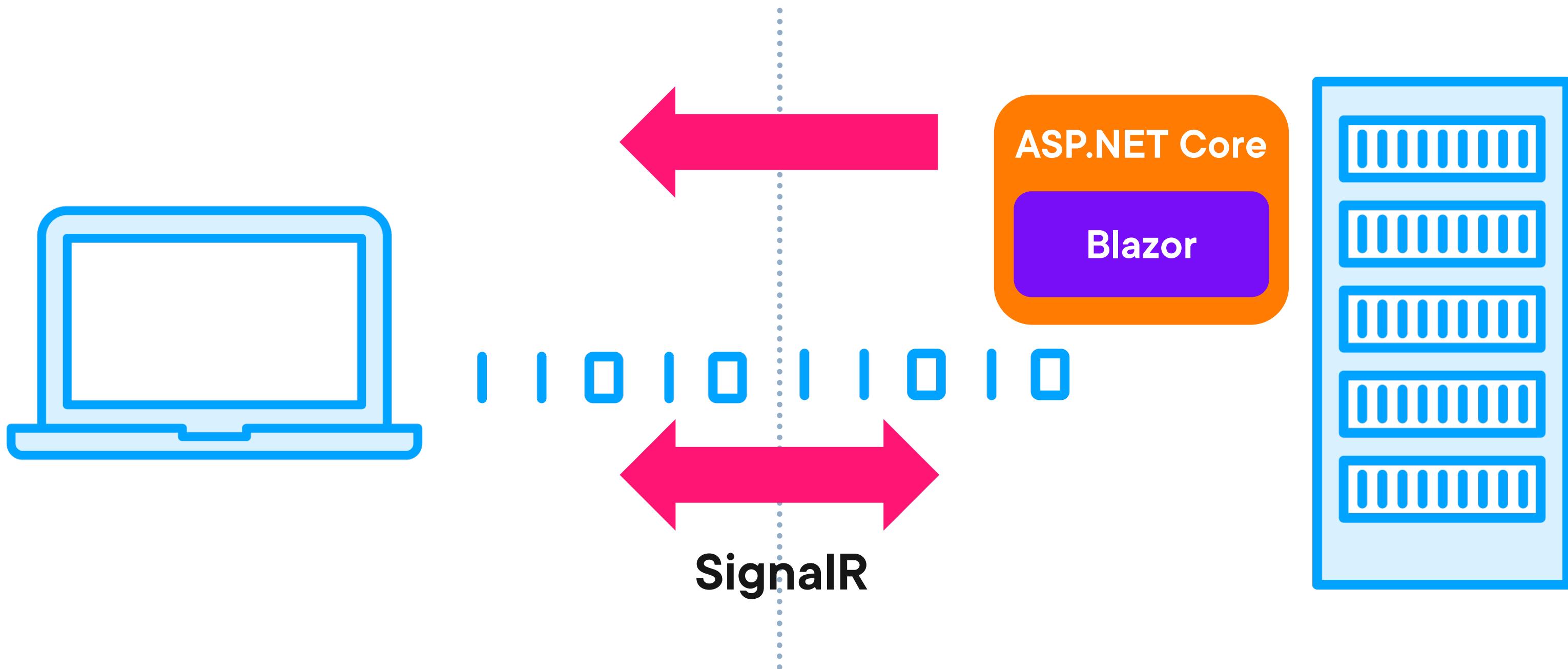
Blazor Hosting Models

Blazor Client
WASM

Blazor Server



Blazor Server



Blazor Hosting Models

Blazor Client
WASM

Blazor Server

Full Stack Web UI
Server-side rendering
Interactivity can be added



ASP.NET Core 6 Blazor Fundamentals

by Gill Cleeren

Blazor is Microsoft's technology to create rich web applications using C# and HTML.

This course will teach you everything you need to know to build a full Blazor application using .NET 6.

 [Resume Course](#)



[Bookmark](#)



[Add to Channel](#)



[Download Course](#)



[Schedule Reminder](#)

Celebrate and see your [Completion Badge or Course Certificate](#)

We can't wait to see what you learn next.

[See Learning Tools](#) ▾

Course author



Gill Cleeren

Gill Cleeren is a Microsoft Regional Director, MVP and Pluralsight author. Gill is the CTO of Xpirit Belgium and focuses on web and mobile architecture. He's also a frequent speaker at many...

Course info

Level **Beginner**

Rating  (10)

My rating 

Duration 5h 42m

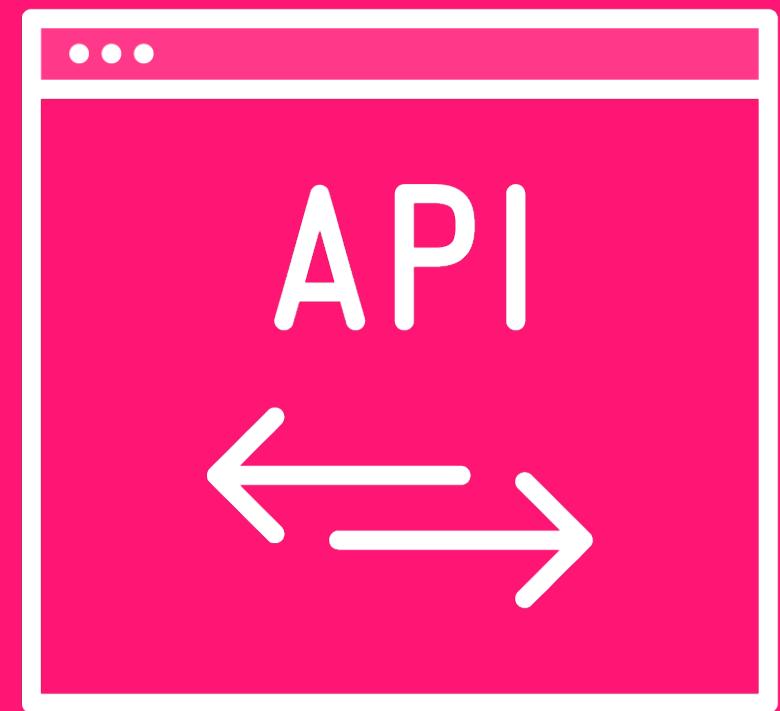
Updated 18 Oct 2022

Share course



Using NSwag and NSwagStudio



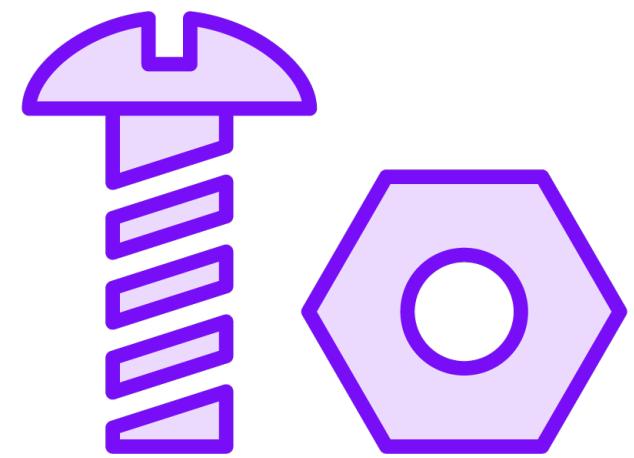


Accessing the API

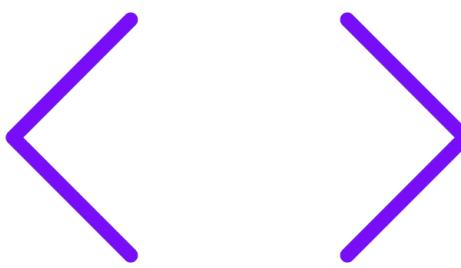
We can write all code manually for each type
of client we'll want to connect.



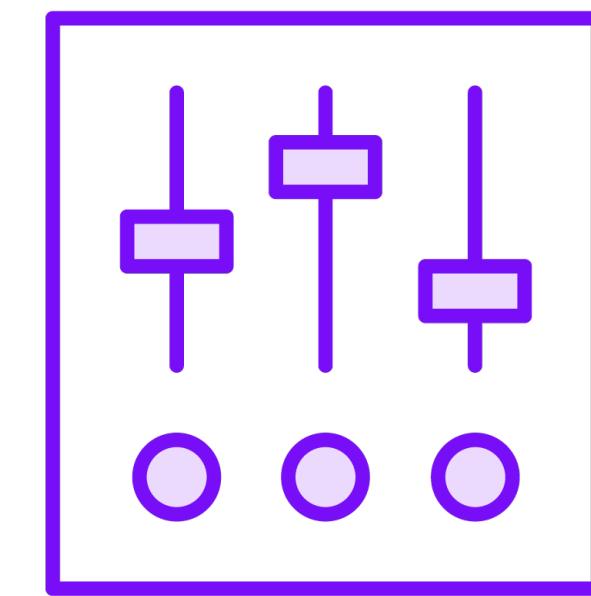
Using NSwag



Tooling based on
Swagger/OpenAPI
specification



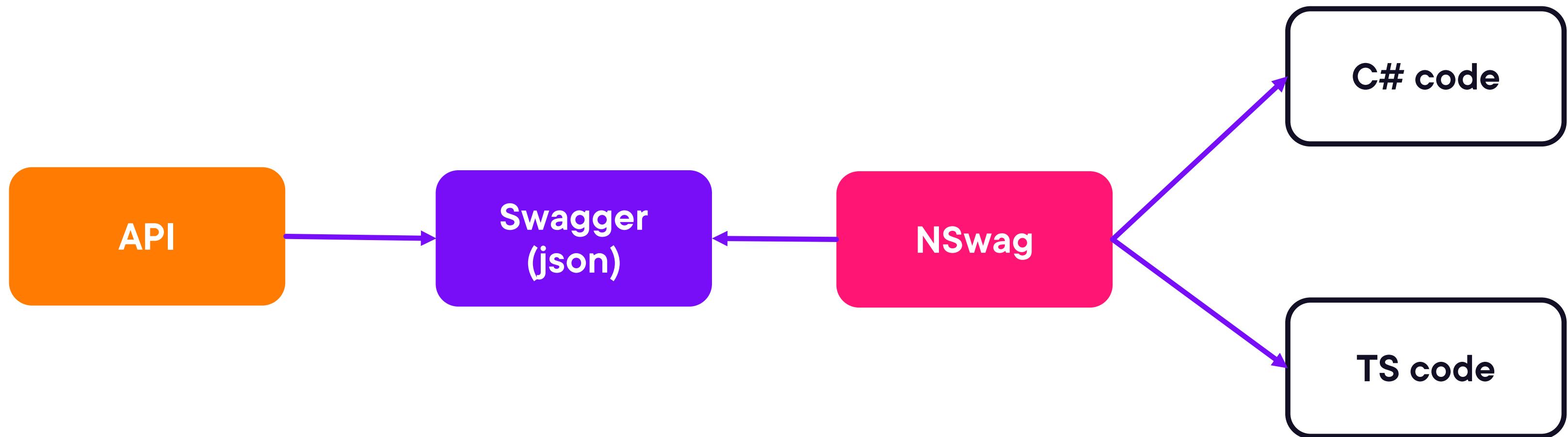
Generation of client
code to use the API

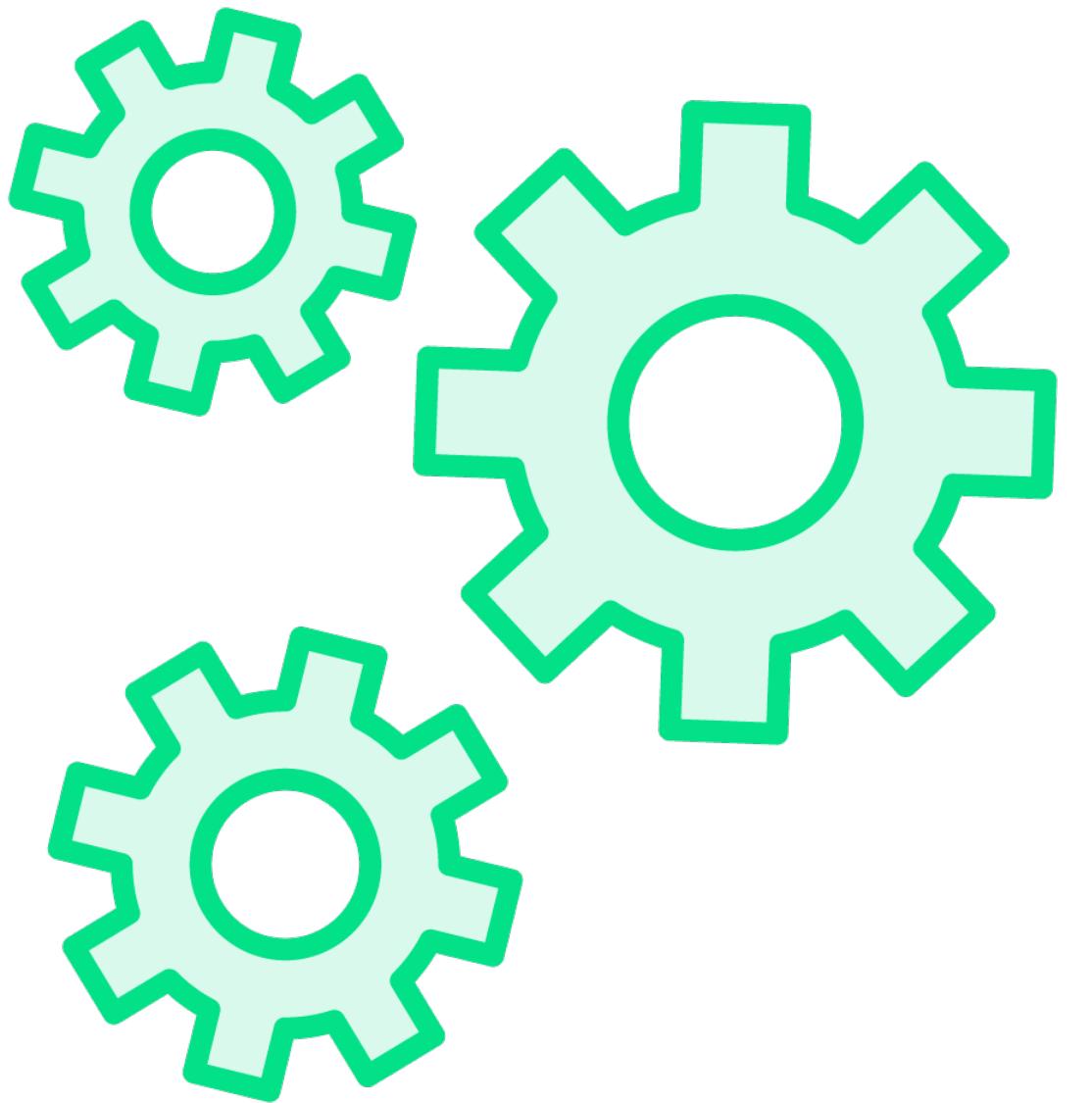


Works for many
technologies
including .NET and
TypeScript



Using NSwag





Using NSwag

- NSwagStudio
- Build
- CLI



NSwagStudio

The screenshot shows the NSwagStudio application window. On the left, the 'Documents' sidebar lists a file named 'nswagsettings.nswag'. The main area is divided into two main sections: 'Input' and 'Outputs'.

Input: OpenAPI/Swagger Specification

- Runtime:** Net60
- Default Variables ('foo=bar,baz=bar'), usage: \${foo}**: A text input field.
- Specification URL:** https://localhost:7020/swagger/v1/swagger.json
- Specification JSON/YAML (if specified, the URL is ignored):** A code editor containing the following JSON code:

```
1{
2  "openapi": "3.0.1",
3  "info": {
4    "title": "GloboTicket Ticket Management API",
5    "version": "v1"
6  },
7  "paths": {
8    "/api/Account/authenticate": {
9      "post": {
10        "tags": [
11          "Account"
12        ],
13        "requestBody": {
14          "content": {
15            "application/json": {
16              "schema": {
17                "$ref": "#/components/schemas/Authenticati
18              }
19            },
20            "text/json": {
21              "schema": {
22                "$ref": "#/components/schemas/Authenticati
23              }
24            }
25          }
26        }
27      }
28    }
29  }
30 }
```

Outputs:

- TypeScript Client CSharp Client CSharp Controller
- OpenAPI/Swagger Specification** (selected) **CSharp Client**
- CSharp Client Settings**
- Namespace:** GloboTicket.TicketManagement.App.Services
- Additional Namespace Usages (comma separated):** A text input field.
- Generate contracts output
- Generate exception classes (when disabled, exception classes must be imported)
- Exception class name (may contain the '{controller}' placeholder):** ApiException
- Client:** Generate Client Classes
- Operation Generation Mode:** MultipleClientsFromOperationId
- Class Name:** {controller}Client
- Client class access modifier:** public
- Methods with a protected access modifier to use in partial methods ('classname.methodname', comma separated):** A text input field.

Buttons at the bottom:

- Generate Outputs
- Generate Files



Demo



Using NSwagStudio to generate client code



Exploring the Client App



The Blazor App

Packages

AutoMapper

HttpClientFactory

NSwag
Service code

Authentication
Covered in next module



Demo



Exploring the Blazor application





**“The list of orders is now
a bit too long.”**

Can we get it paged?”



Demo



Adding paging functionality on the API

Generating new code with NSwag

Updating the Blazor app



Summary



NSwag works well with Swagger API

- Less code to write

Blazor can plug in on our architecture



Up Next:

Handling cross-cutting concerns

