
OpenAPI Client (Swagger)

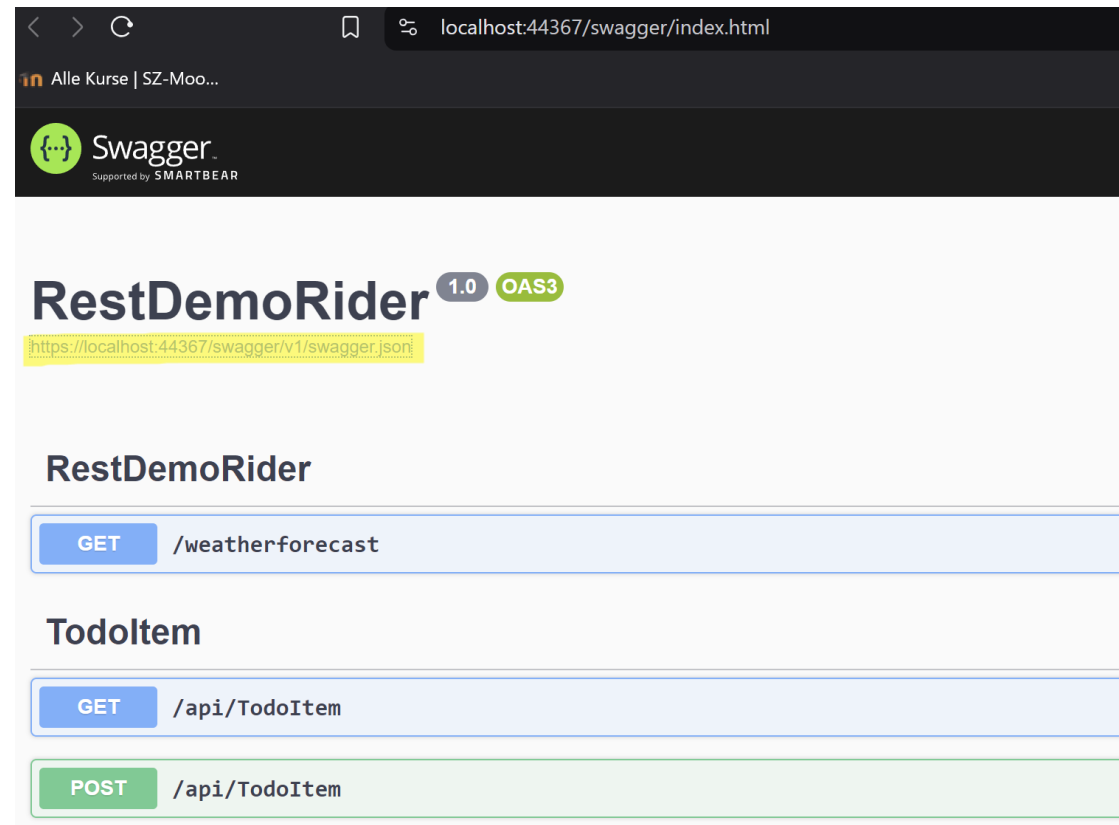
PROF. RALPH JANK, MSC

Allgemeines Open API

- OpenAPI ist eine Open-Source-Spezifikation, die von der OpenAPI Initiative entwickelt wurde, um APIs öffentlich zugänglich zu machen.
- **Standardisierte Beschreibung:** OpenAPI bietet eine standardisierte Sprache, um APIs zu beschreiben → Kommunikation zwischen Entwicklern wird erleichtert
- **API-Dokumentation:** OpenAPI ermöglicht die Erstellung von detaillierten API-Dokumentationen
- **Interoperabilität:** OpenAPI ermöglicht die Interoperabilität zwischen verschiedenen Plattformen und Sprachen
- **Automatisierte API-Generierung:** OpenAPI kann verwendet werden, um APIs automatisch zu generieren

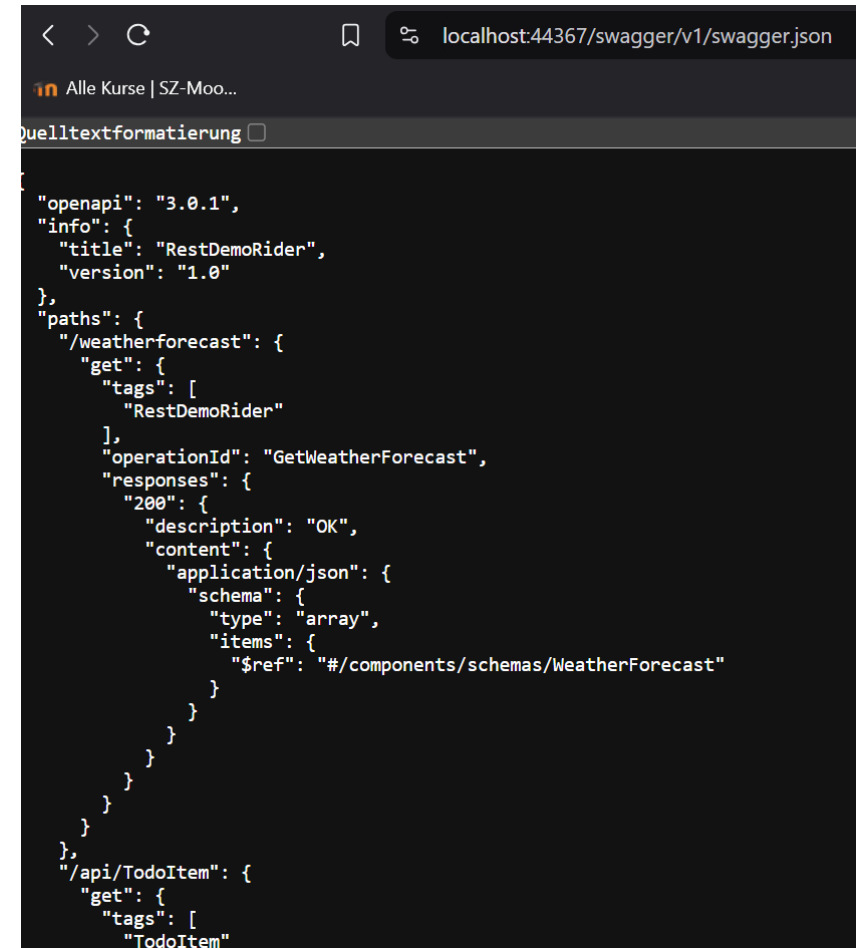
Rest Service (Swagger) Spezifikation I

- Service ausführen
- Open API Spezifikation ist im angeführten Link enthalten



Rest Service (Swagger) Spezifikation II

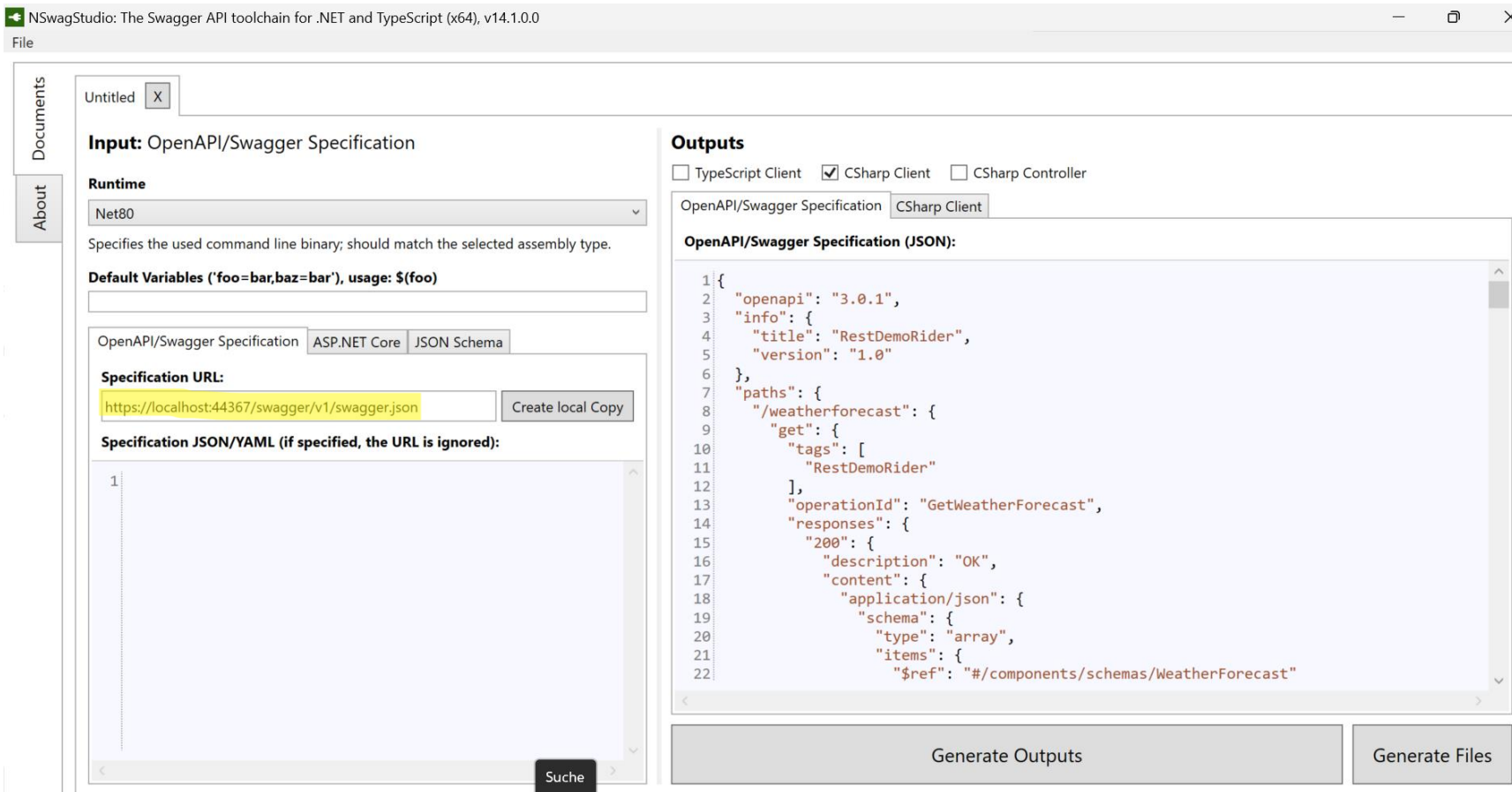
- Gesamte Service Spezifikation in Json verfügbar
- Kann für die Client-Erstellung genutzt werden



The screenshot shows a web browser window with the address bar displaying `localhost:44367/swagger/v1/swagger.json`. The page content is a JSON specification for a REST API. The visible portion of the JSON includes the `openapi` version, `info` object with title and version, and the start of the `paths` object. The first path is `/weatherforecast` with a `get` method. The `tags` array contains `RestDemoRider`. The `operationId` is `GetWeatherForecast`. The `responses` object shows a `200` status with a description of "OK" and a content type of `application/json`. The content schema is an array of items, each with a `$ref` pointing to `#/components/schemas/WeatherForecast`. The JSON is partially visible, showing the structure for the `weatherforecast` endpoint and the beginning of the `api/ToDoItem` endpoint.

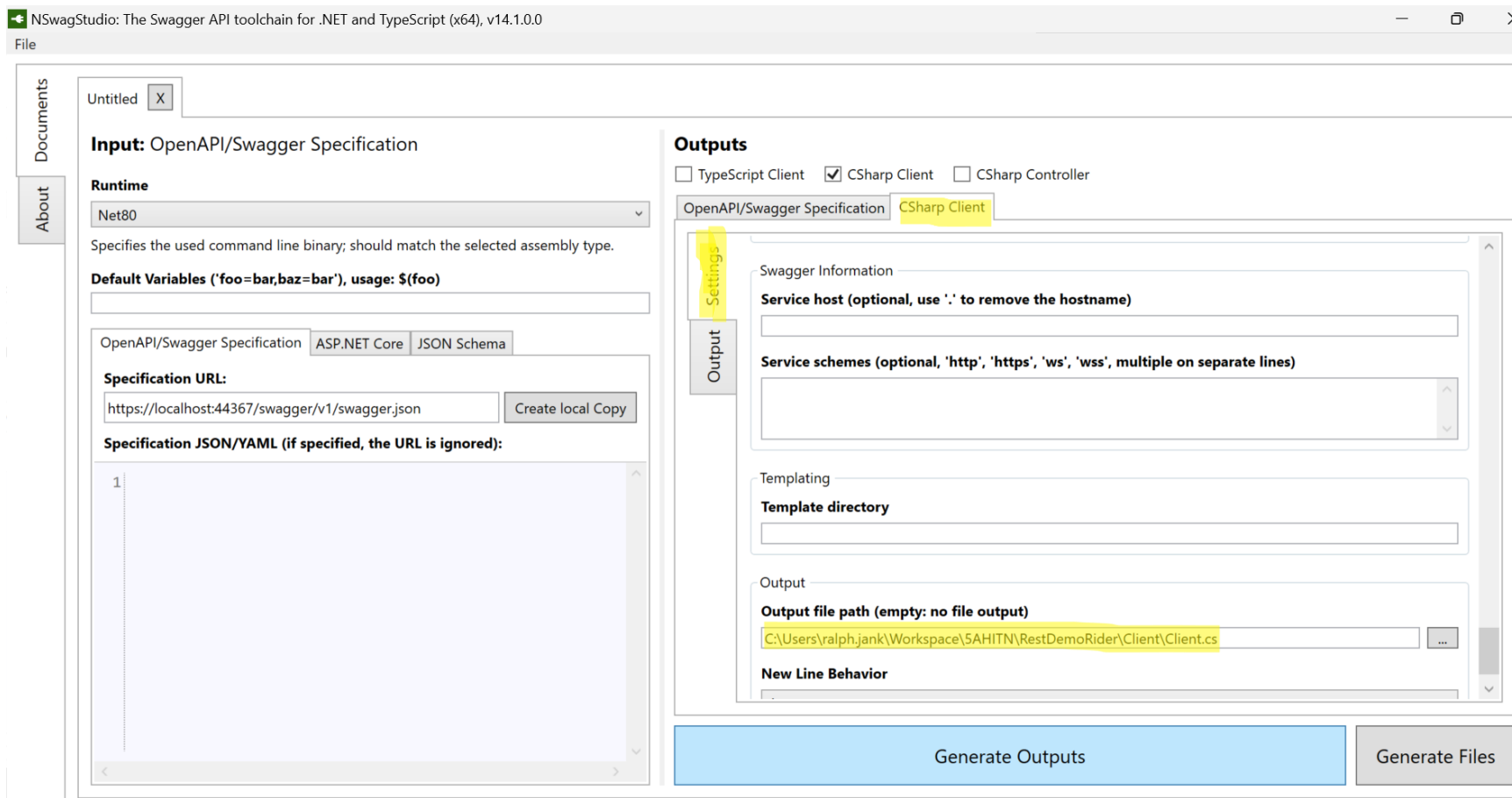
```
{
  "openapi": "3.0.1",
  "info": {
    "title": "RestDemoRider",
    "version": "1.0"
  },
  "paths": {
    "/weatherforecast": {
      "get": {
        "tags": [
          "RestDemoRider"
        ],
        "operationId": "GetWeatherForecast",
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "application/json": {
                "schema": {
                  "type": "array",
                  "items": {
                    "$ref": "#/components/schemas/WeatherForecast"
                  }
                }
              }
            }
          }
        }
      }
    },
    "/api/ToDoItem": {
      "get": {
        "tags": [
          "ToDoItem"
        ]
      }
    }
  }
}
```

NSwagStudio I



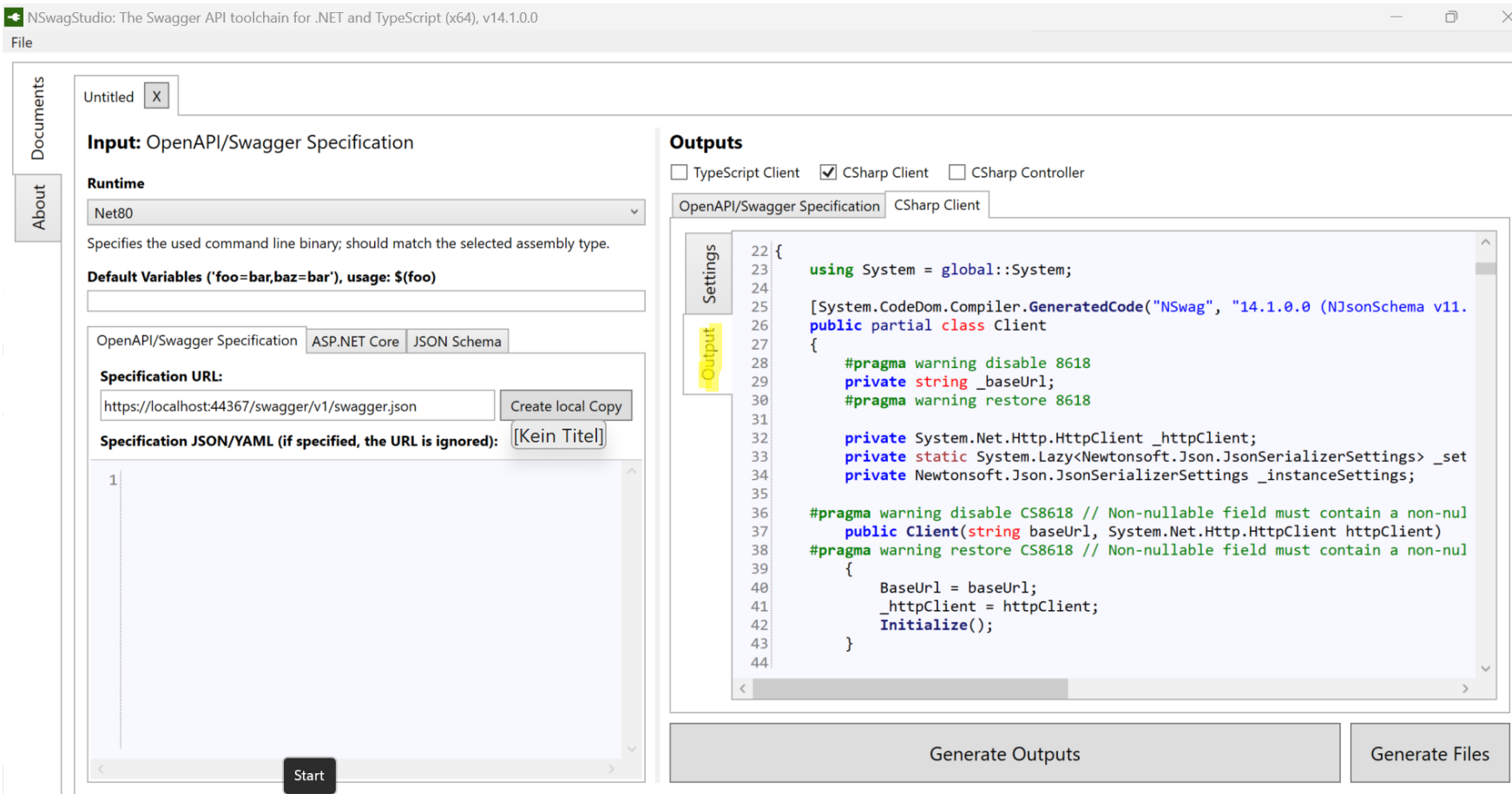
- NSwagStudio kann zum Generieren von Client und C# Controller verwendet werden
- Zum Generieren des Clients ist die „Specification URL“ zu setzen (URL aus Swagger Rest API)

NSwagStudio II



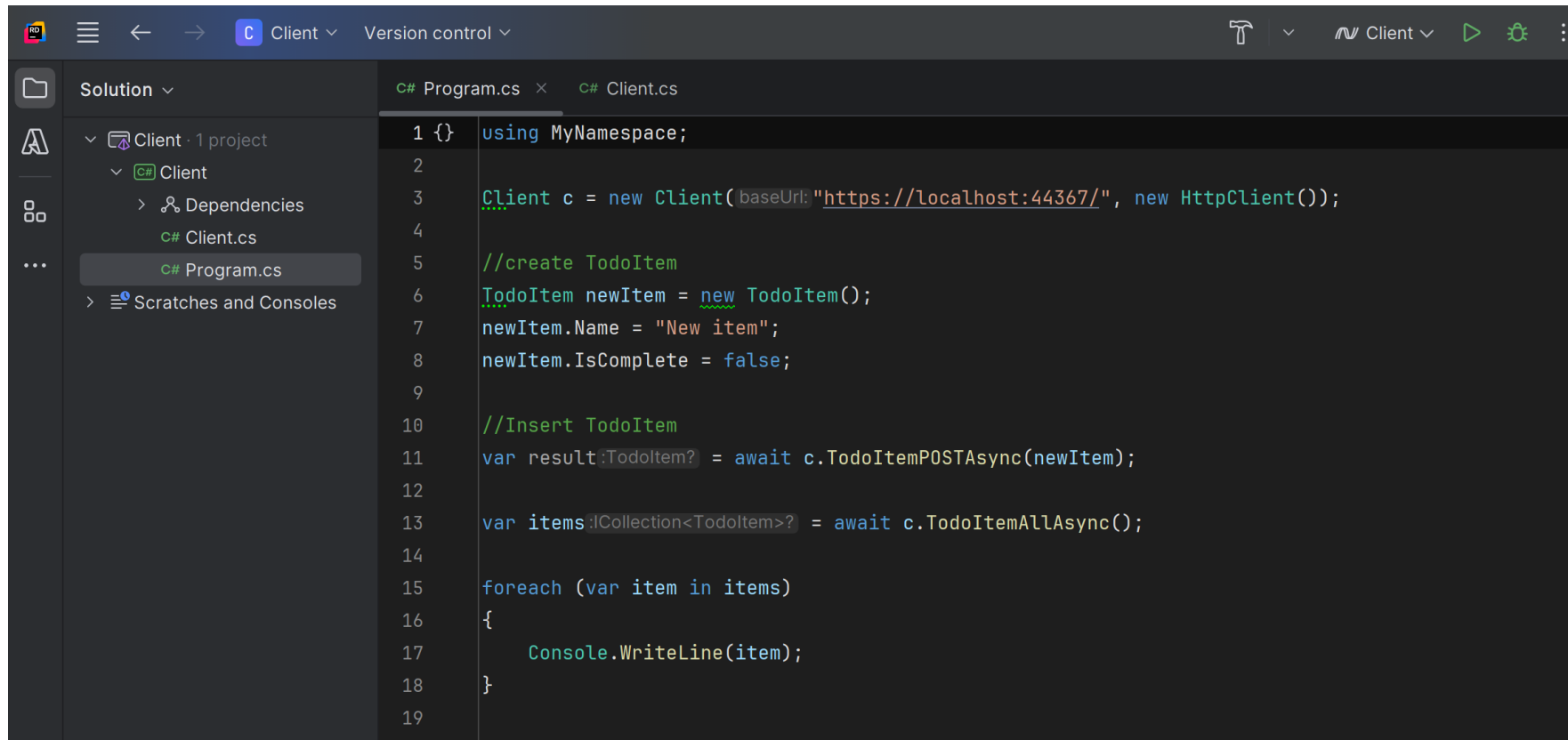
- Unter dem Tab „Csharp Client/Settings“ ist die Ausgabedatei zu setzen
- Falls keine Datei gesetzt ist, wird **keine Ausgabe erzeugt**

NSwagStudio III



- Unter dem Tab „Csharp Client/Output“ kann der erstellte Output überprüft werden

Client in Konsolenanwendung testen



```
1 {} using MyNamespace;
2
3 Client c = new Client(baseUrl: "https://localhost:44367/", new HttpClient());
4
5 //create TodoItem
6 TodoItem newItem = new TodoItem();
7 newItem.Name = "New item";
8 newItem.IsComplete = false;
9
10 //Insert TodoItem
11 var result:TodoItem? = await c.TODOItemPOSTAsync(newItem);
12
13 var items:ICollection<TodoItem>? = await c.TODOItemAllAsync();
14
15 foreach (var item in items)
16 {
17     Console.WriteLine(item);
18 }
19
```

- Neue Konsolenanwendung anlegen
- Erstellte Datei (Client.cs) einbinden
- Nuget Package Newtonsoft.Json hinzufügen
- Client mit BaseURL anlegen

Nuget Package Newtonsoft.Json hinzufügen

