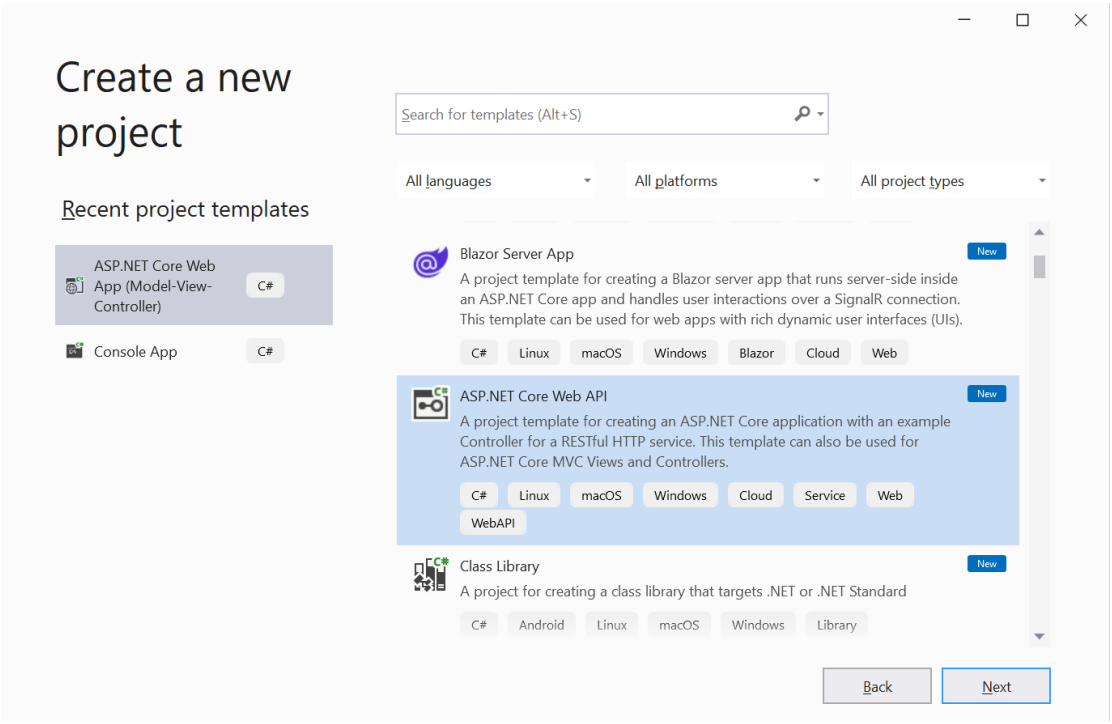
## *Ajax, jQuery, ASP.NET Web API*

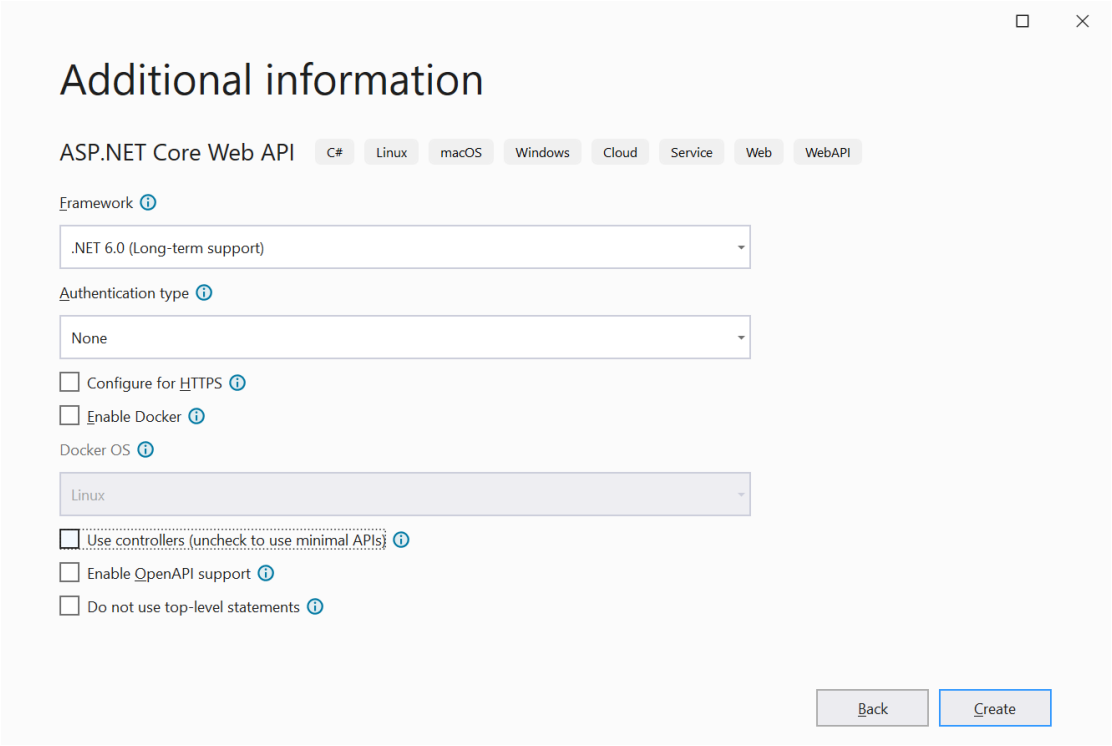
Do realizacji ćwiczeń potrzebne jest zintegrowane środowisko programistyczne Microsoft Visual Studio 2022.

Celem ćwiczeń jest wykorzystanie techniki Ajax do komunikacji z API zaimplementowaną w ASP.NET Web API (w architekturze Minimal API) z poziomu strony HTML w przeglądarce. Usługa będzie obliczać wyniki dodawania, odejmowania, mnożenia i dzielenia dla podanych dwóch liczb całkowitych.

1. Create a new project (Project).
   a) Uruchom narzędzie Microsoft Visual Studio.
   b) Z menu głównego wybierz File→New→Project (lub Create a new project z ekranu startowego). Wybierz szablon ASP.NET Core Web API. Kliknij przycisk Next. Zmień proponowaną nazwę projektu na „WebApiAjax”. Zaakceptuj zaproponowany katalog lub zmień go na inny gdy nie masz prawa zapisu w proponowanym katalogu. Pozostałe opcje pozostaw domyślne. Kliknij przycisk Next.




   c) W oknie wyboru wersji frameworka .NET wybierz .NET 6.0 (Long-term support). Odznacz pole wyboru Use Controllers, gdyż wykorzystamy Minimal API. Odznacz pole wyboru Enable OpenAPI support, gdyż nasze API będzie tak proste, że Swagger do jego testowania nie będzie konieczny. Odznacz pole wyboru Configure for HTTPS. Pozostałe opcje pozostaw domyślne i kliknij przycisk Create.




2. Z poziomu węzła projektu wybierz opcję Add i utwórz w głównym folderze nową klasę. Nazwij ją „CalcResult". Klasa ta będzie reprezentować strukturę z wynikami obliczeń, którą Web API będzie serializować do formatu JSON. Jako ciało klasy wklej poniższy kod:

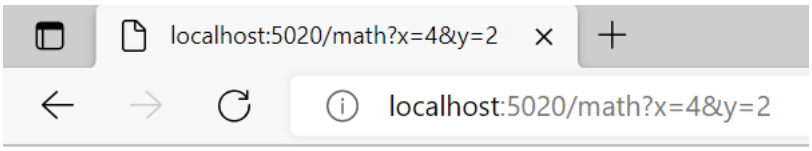
```
public class CalcResult
{
    public int Sum { get; set; }
    public int Difference { get; set; }
    public int Product { get; set; }
    public int Quotient { get; set; }
}
```

3. W pliku Program.cs przed instrukcją uruchamiającą aplikację wstaw poniższą instrukcję wiążącą trasę (adres URI i metodę protokołu HTTP) z funkcją obsługi żądania. Dodaj odpowiednią instrukcję using.

```
app.MapGet("/math",
    (int x, int y) => new CalcResult() { Sum = x+y, Difference = x-y,
                                         Product = x*y, Quotient = x/y });
```

Wynikiem działania funkcji lambda jest obiekt przygotowanej wcześniej klasy CalcResult. Jego serializacją do formatu JSON zajmie się framework ASP.NET. ASP.NET obsłuży również przekazanie parametrów zawartych w żądaniu HTTP jako parametrów funkcji lambda. Ponieważ parametry x i y nie zostały zdefiniowane w atrybutach routingu, framework przyjmie, że te parametry będą przekazywane w ciągu znaków zapytania (ang. query string).

4. Uruchom aplikację kombinacją klawiszy Ctrl+F5 (Start Without Debugging). W przeglądarce powinien zostać wyświetlony efekt działania przykładowej trasy, która została zdefiniowana przez kreator projektu (prognoza pogody w formacie JSON). Popraw adres URI na taki, który wywoła nasz kontroler API, przekazując w adresie parę wartości całkowitoliczbowych dla parametrów x i y. W rezultacie przeglądarka powinna wyświetlić dokument JSON zawierający wyniki obliczeń, tak jak na przedstawionym poniżej obrazku (oczywiście numer portu oraz konkretne wartości liczbowe mogą być inne).



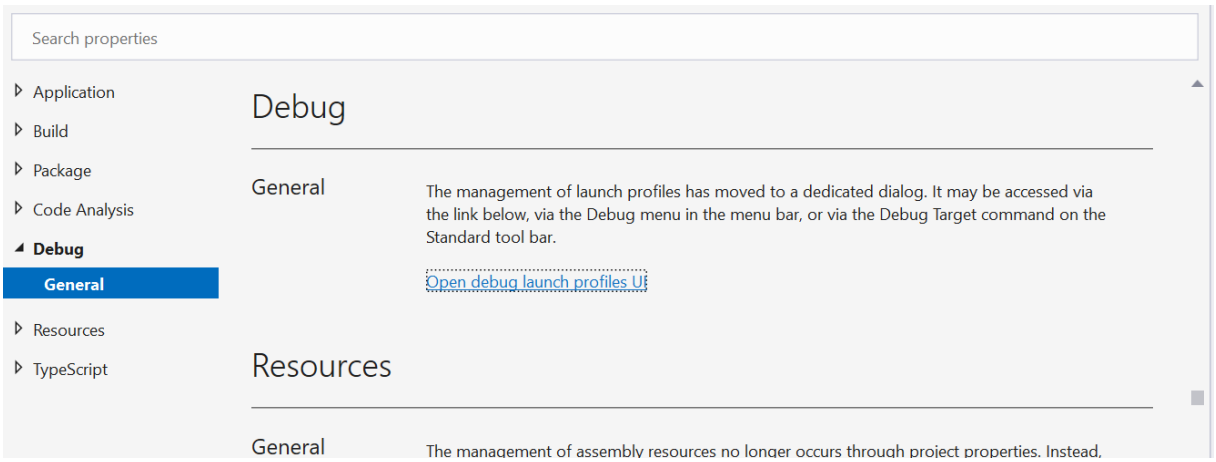

```
{"sum":6,"difference":2,"product":8,"quotient":2}
```

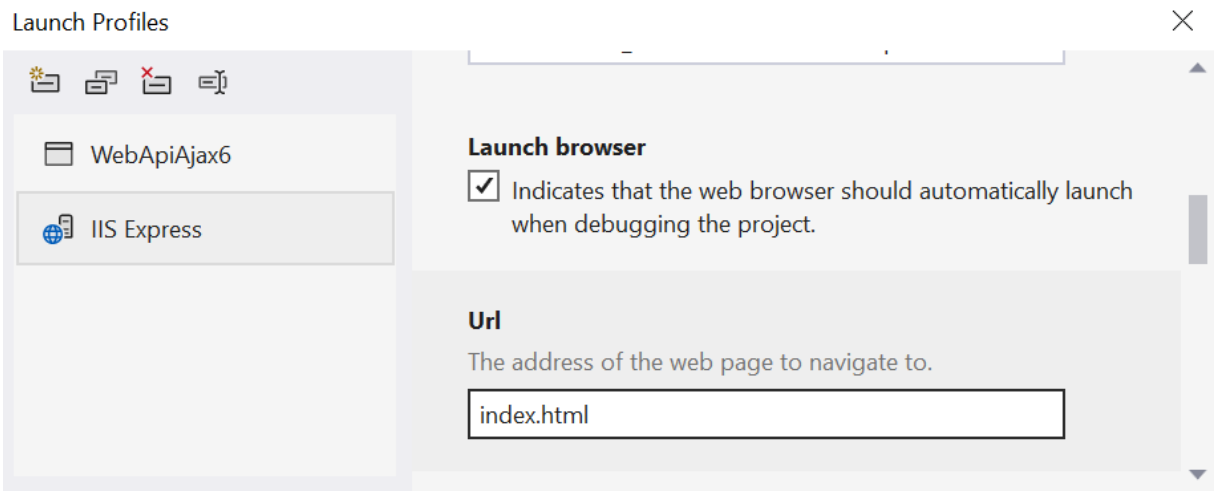
5. Dodaj do projektu statyczną startową stronę HTML aplikacji. Ponieważ projekty API nie są domyślnie skonfigurowane do serwowania statycznej zawartości, konieczne będzie wykonanie kilku kroków konfiguracji:

a) Kliknij prawym klawiszem myszy na węźle projektu w panelu Solution Explorer (poniżej węzła rozwiązania) i wybierz opcję Add→New Folder z menu kontekstowego. Nazwij folder „wwwroot”.
b) Kliknij prawym klawiszem myszy w panelu Solution Explorer na nowo utworzonym folderze wwwroot i wybierz opcję Add→New Item z menu kontekstowego. Wybierz stronę HTML jako typ tworzonego elementu. Nazwij stronę „index.html”.
W pliku Program.cs bezpośrednio przed instrukcją uruchamiającą aplikację wstaw poniższą instrukcję:

```
app.UseStaticFiles();
```

c) Otwórz stronę index.html do edycji i umieść w jej sekcji <body> dowolny tekst.
d) Wyświetl właściwości projektu klikając prawym klawiszem myszy na węźle projektu w panelu Solution Explorer i wybierając opcję Properties z menu kontekstowego.
e) W sekcji Debug właściwości projektu kliknij link Open debug launch profiles UI, a następnie w wyświetlonym oknie dialogowym zmodyfikuj oba dostępne profile wpisując „index.html” w polu Url.






f) Zapisz wszystkie zmiany i przebuduj projekt (Build).
g) Uruchom aplikację aby sprawdzić czy nowa strona startowa wyświetli się w przeglądarce.

6. W kodzie startowej strony HTML umieść poniższy element <script> w sekcji <head>, dołączający bibliotekę jQuery.

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
```

7. Zastąp zawartość sekcji <body> poniższym kodem.

```
    <script type="text/javascript">
    $(document).ready(function() {
      $("#calc").click(function () {
        var x = ...;
        var y = ...;

        $.ajax(...);
      });
    });
    </script>
    <div>
        <h1>Calculations</h1>
        <form>
            x = <input type="text" id="x" />
            y = <input type="text" id="y" />
            <input type="button" id="calc" value="Calculate" />
        </form>
        x + y = <span id="sum"></span><br />
        x - y = <span id="difference"></span><br />
        x * y = <span id="product"></span><br />
        x / y = <span id="quotient"></span><br />
    </div>
```

Możesz teraz odświeżyć stronę w przeglądarce aby obejrzeć jej aktualny wygląd. Nie klikaj jeszcze przycisku na stronie, gdyż logika JavaScript strony nie została jeszcze w pełni zaimplementowana.

8. W miejscu wielokropków samodzielnie dopisz kod realizujący następujące zadania (wykorzystując jQuery):

a) odczyt wartości wprowadzonych do pól formularza i ich zapisanie w przygotowanych zmiennych
b) wysłanie żądania Ajax i umieszczenie odebranych wyników obliczeń w przygotowanych elementach  (jako adres URI dla żądania podaj „/math").

9. Przetestuj działanie aplikacji dla różnych par wartości całkowitoliczbowych.

## Zadanie do samodzielnego wykonania

1. Zmień etykietę przycisku na „Call API (jQuery)".

2. Dodaj obok istniejącego przycisku drugi przycisk z etykietą „Call API (vanilla JS)".

3. Samodzielnie zaimplementuj obsługę dodanego przycisku tak aby jego działanie funkcjonalnie było takie samo jak pierwszego przycisku, ale aby implementacja była w czystym języku JavaScript (bez jQuery).

- do wysłania żądania do API wykorzystaj bezpośrednio obiekt XMLHttpRequest

- pamiętaj o ustawieniu nagłówka dla wysyłanego żądania, tak aby serwer odpowiedział danymi w formacie JSON

- do odczytania parametrów z pól formularza i modyfikacji elementów przygotowanych do wyświetlania wyników obliczeń wykorzystaj interfejs DOM HTML-a z poziomu czystego JavaScriptu