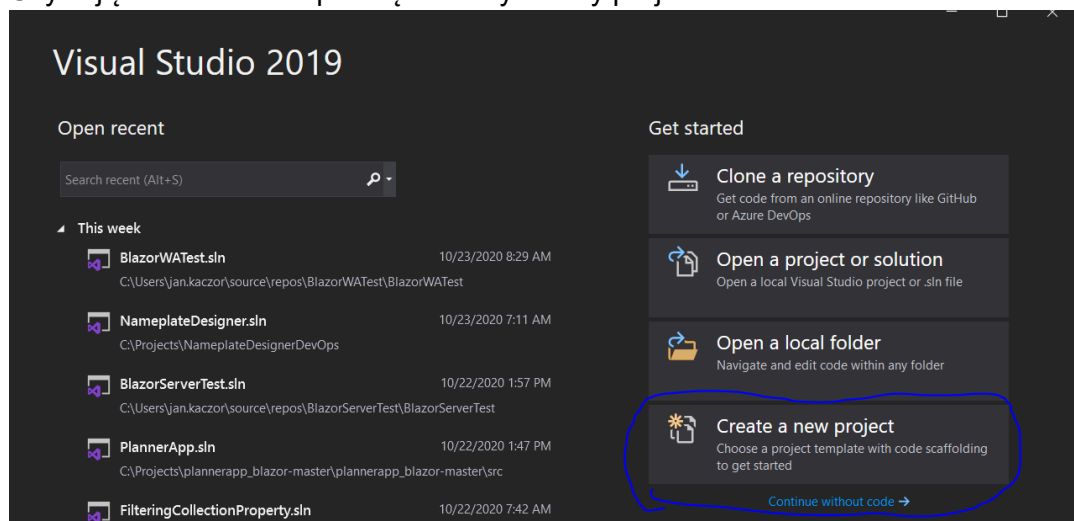
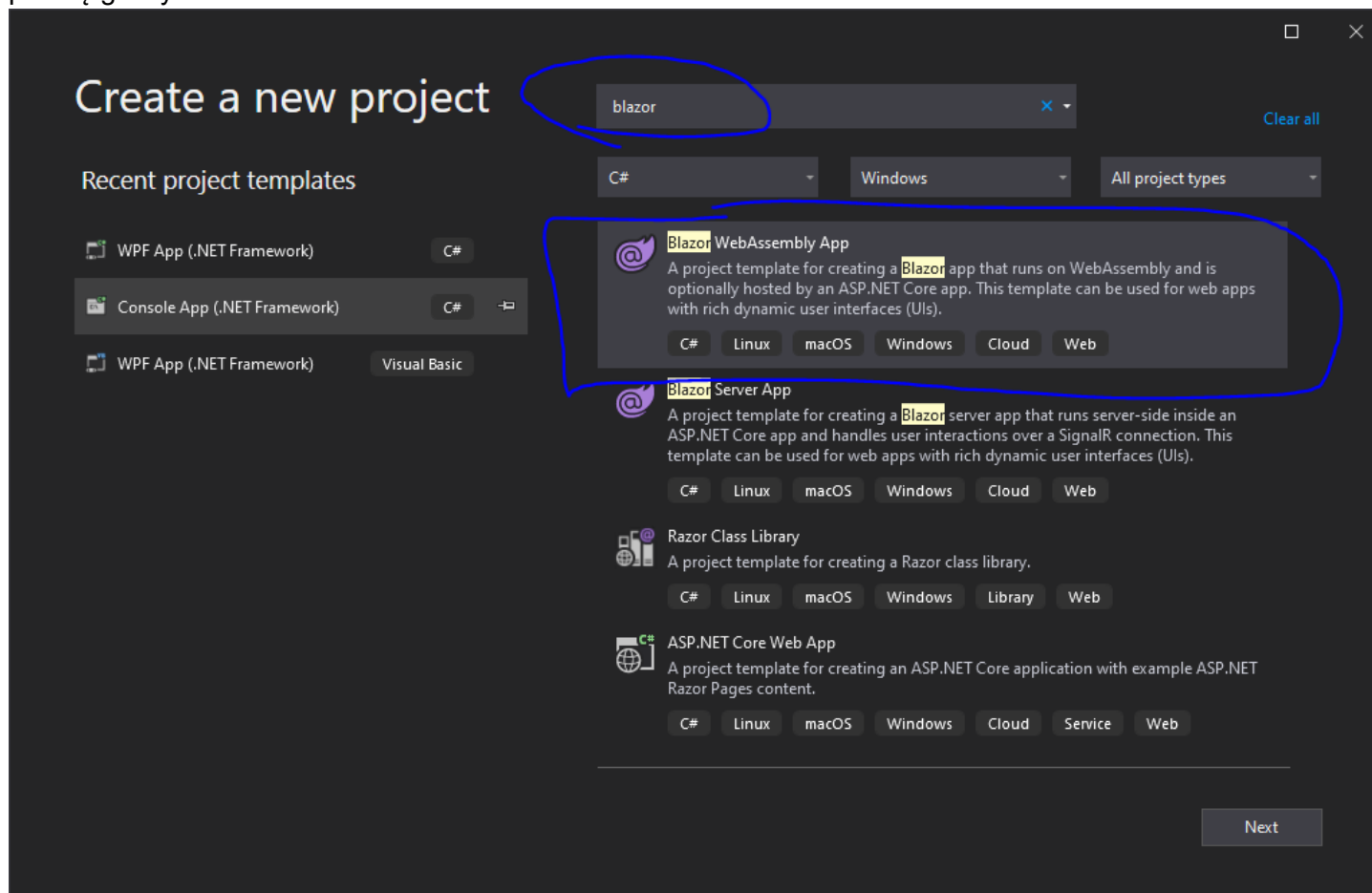


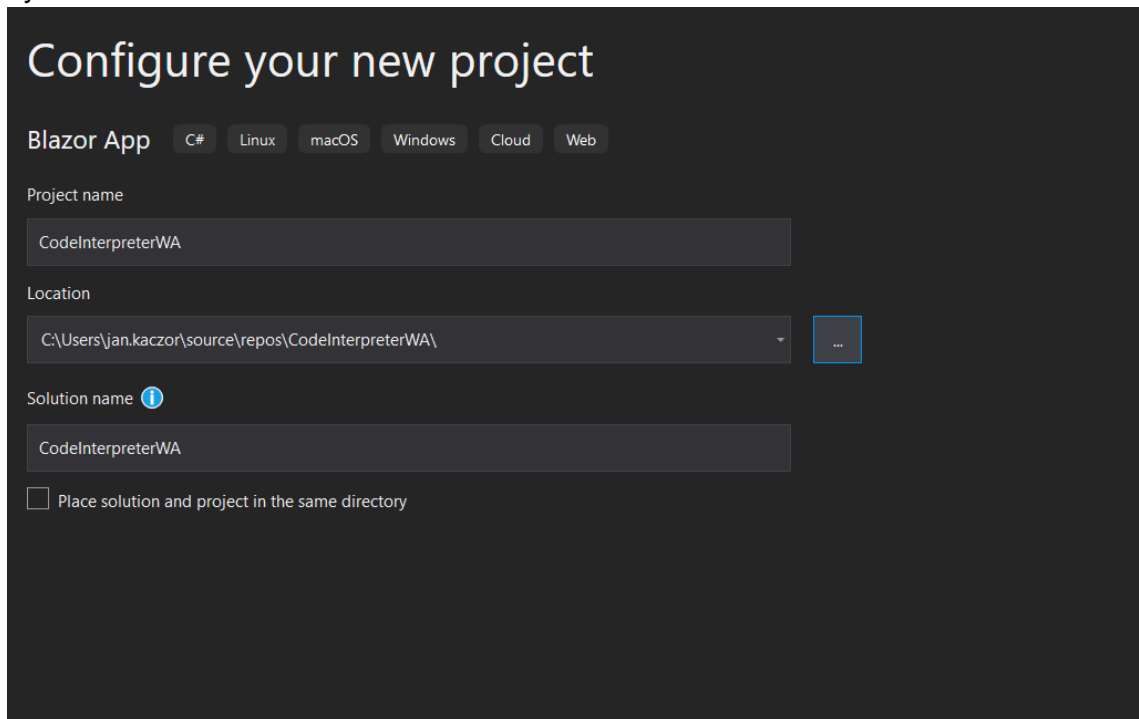
1. Używając Visual studio proszę utworzyć nowy projekt:



2. W wyszukiwarce proszę wpisać *blazor* powinien się pojawić template **Blazor WebAssembly App**, proszę go wybrać.



3. Następnie proszę podać w polu nazwa projektu wpisać **CodeInterpreterWA** oraz wybrać miejsce na dysku.



Configure your new project

Blazor App C# Linux macOS Windows Cloud Web

Project name

CodeInterpreterWA

Location

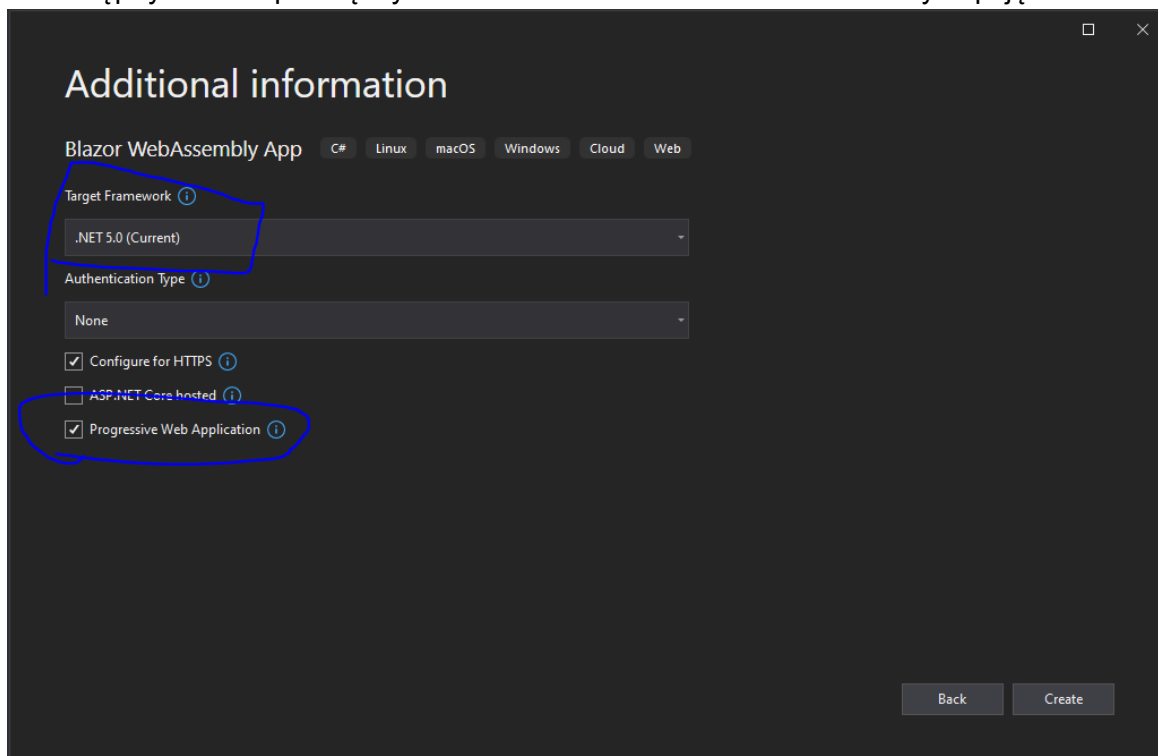
C:\Users\jan.kaczor\source\repos\CodeInterpreterWA\

Solution name ⓘ

CodeInterpreterWA

☐ Place solution and project in the same directory

4. W następnym kroku proszę wybrać framework .NET 5.0 oraz zaznaczyć opcję PWA i kliknąć Create.



Additional information

Blazor WebAssembly App C# Linux macOS Windows Cloud Web

Target Framework ⓘ

.NET 5.0 (Current)

Authentication Type ⓘ

None

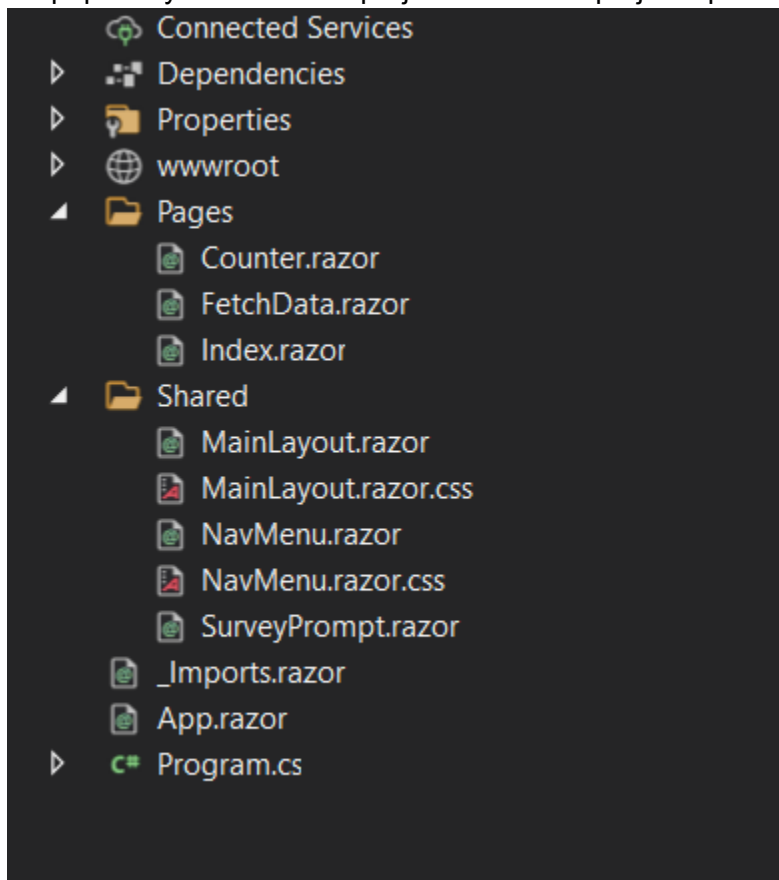
☒ Configure for HTTPS ⓘ

☐ ASP.NET Core hosted ⓘ

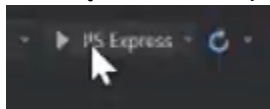
☒ Progressive Web Application ⓘ

Back Create

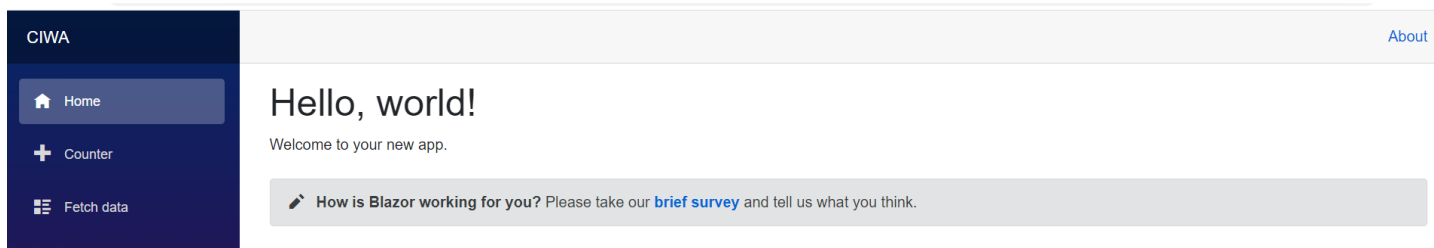
5. Po poprawnym utworzeniu projektu struktura projektu powinna się prezentować w następujący sposób:



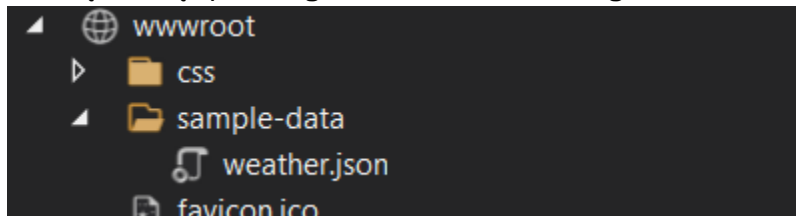
6. Proszę uruchomić projekt klikając F5 albo przycisk



7. Powinna załadować się strona która wygląda następująco:



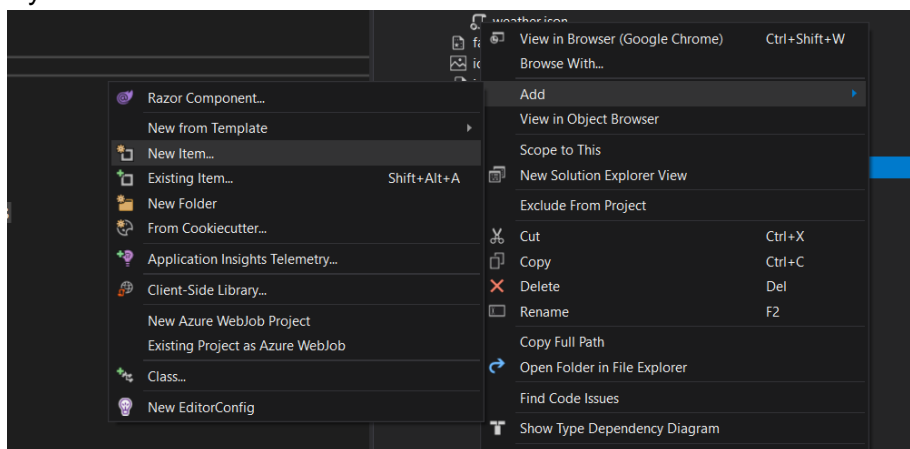
8. Proszę usunąć pliki **Pages/Counter.razor**, **Pages/FetchData.razor** oraz folder **sample-data** z solucji.



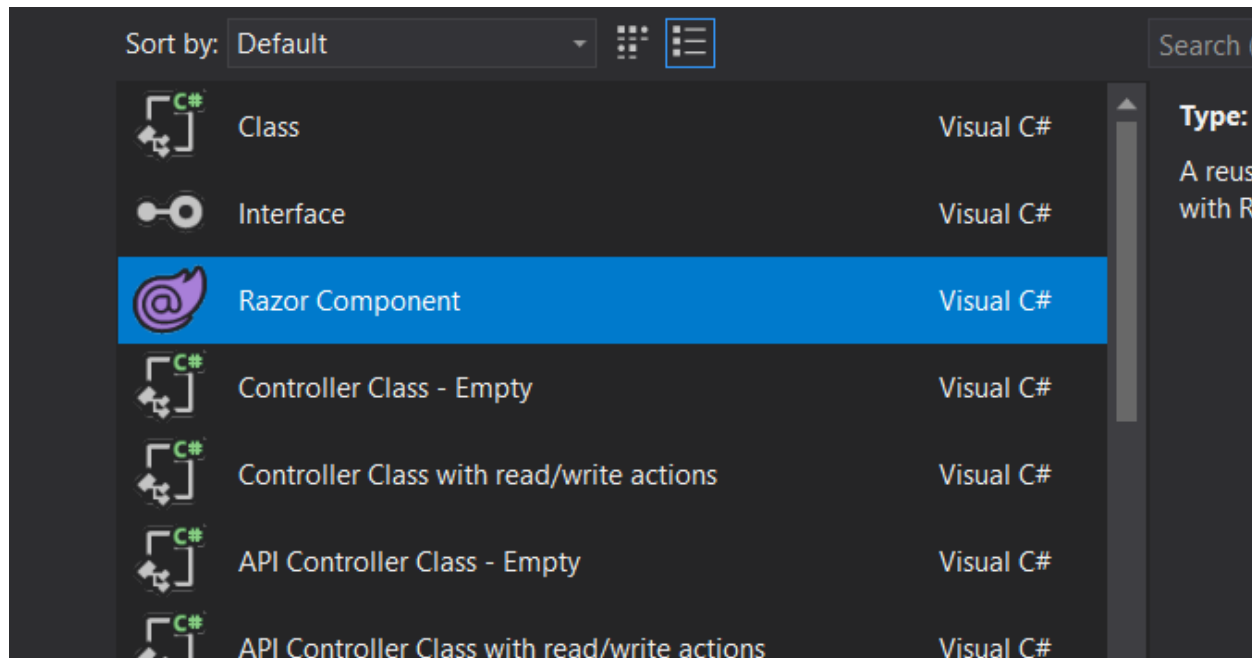
9. Proszę zaktualizować plik **shared/NavMenu.razor**, jest on odpowiedzialny za nawigację w naszej aplikacji. W tym pliku w dalszym ciągu znajdują się nawigacje do komponentów które przed chwilą usunęliśmy **Counter** i **FetchData**. Usuń kod zaznaczony poniżej.

```
<div class="@NavMenuCssClass" @onclick="ToggleNavMenu">
  <ul class="nav flex-column">
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="" Match="NavLinkMatch.All">
        <span class="oi oi-home" aria-hidden="true"></span> Home
      </NavLink>
    </li>
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="counter">
        <span class="oi oi-plus" aria-hidden="true"></span> Counter
      </NavLink>
    </li>
    <li class="nav-item px-3">
      <NavLink class="nav-link" href="fetchdata">
        <span class="oi oi-list-rich" aria-hidden="true"></span> Fetch data
      </NavLink>
    </li>
  </ul>
</div>
```

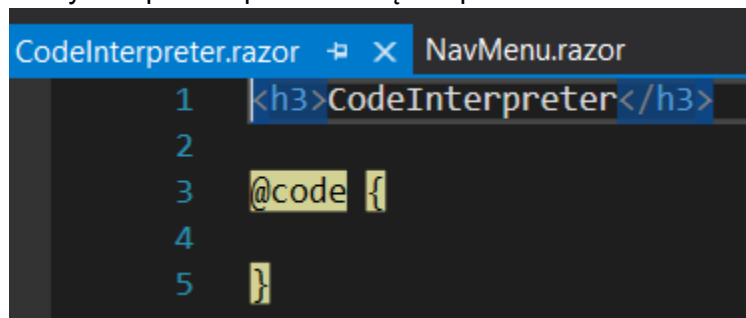
10. Dodaj nowy komponent. W tym celu na folderze **Pages** kliknij prawym przyciskiem myszy a następnie wybierz **Add>New Item...**



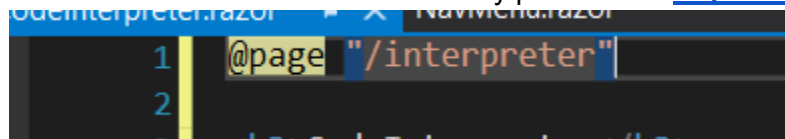
11. W nowym oknie proszę wybrać Razor Component i nazwać go **CodeInterpreter** a następnie kliknąć **Add**



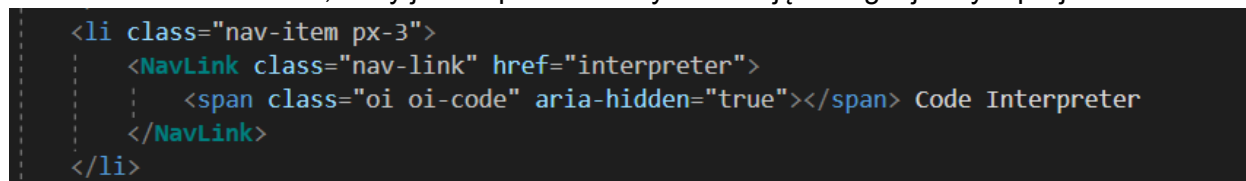
12. Nowy komponent powinien się tak prezentować:



13. W pierwszej linii proszę dodać routing do strony dodając następującą dyrektywę **@page "/interpreter"**. Pozwoli nam to na załadowanie strony pod linkiem <https://localhost:port/interpreter>.



14. Proszę dodać component jako jeden z elementów nawigacji, w tym celu proszę zmodyfikować plik **shared/NavMenu.razor**, który jest odpowiedzialny za sekcję nawigacji w tym projekcie.



15. Proszę sprawdzić czy component działa. W tym celu należy uruchomić aplikację następnie z poziomu nawigacji po lewej stronie wybrać *Code Interpreter*.

16. Następnie proszę edytować kod komponentu **CodeInterpreter.razor** aby wyglądał następująco

```
@page "/"/interpreter"

<h1>Interpreter</h1>

<div class="row" style="height: 600px;">
  <div class="col-6 h-100">
    <textarea class="form-control h-100 "></textarea>
  </div>
  <div class="col-6 h-100">
  </div>
</div>

@code {
}
```

17. Od tego momentu na naszej stronie mamy element textarea który będziemy wykorzystywać w dalszej części kroków jako edytor. Pierwszą rzeczą którą musimy zrobić jest przechwycenie tekstu wpisanego do textarea przez użytkownika. W tym celu w sekcji **@code {}** proszę zdefiniować zmienną typu string i przypisać do niej pusty ciąg znaków. Przykład:

```
@code {

    string _markdownCode = string.Empty;

}
```

18. Następnie nowo utworzoną zmienną musimy powiązać z textarea, aby to co użytkownik wpisze w textarea zostało przekazane do zmiennej. W tym celu użyjemy wiązania danych (**@bind**). [ASP.NET Core Blazor powiązania danych](#). Proszę do elementu textarea dodać następujący kod

```
<textarea @bind="_markdownCode" >
```

Pamiętając aby użyć odpowiedniej nazwy zmiennej.

19. Kolejnym krokiem jest wyświetlenie tekstu przypisanego do zmiennej w tym celu musimy wyświetlić wartość naszej zmiennej w kodzie html.

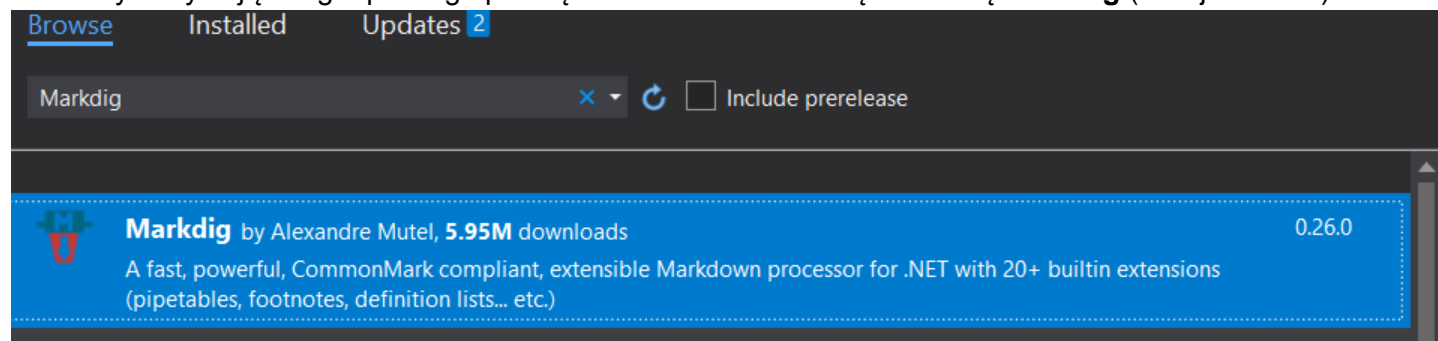
Pomiędzy znacznikami div proszę umieścić zmienną zdefiniowaną w kroku 17.

```
<div class="col-6 h-100">
  @_markdownCode
</div>
```

20. Od tego momentu to co wpisujemy w textarea powinno się wyświetlić z prawej strony. Binding wykonuje się domyślnie na zdarzeniu **onchange** oznacza to że tekst wyświetli się nam dopiero kiedy stracimy focus na elemencie textarea. Nie jest to zachowanie którego oczekujemy w naszym komponencie ponieważ chcemy żeby kod html wyświetlał się od razu przy każdym naciśnięciu klawisza. Proszę zmienić domyślne **onchange** na **oninput** pozwoli to nam na przechwycenie tekstu przy każdej zmianie wartości pola tekstowego. Inaczej mówiąc za każdym razem kiedy naciśniemy klawisz zmienna `_markdownCode` zostanie zaktualizowana.

```
<textarea @bind="_markdownCode" @bind:event="oninput">
```

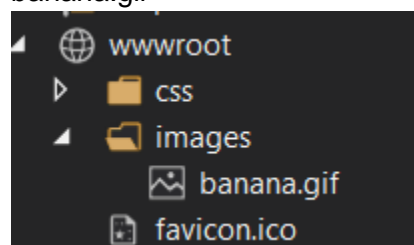
21. Teraz wykorzystując nuget package proszę zainstalować .net'ową bibliotekę **Markdig** (wersja: 0.26.0)



22. Zamiast wyświetlać zwykły tekst identyczny z tym wpisanym przez użytkownika chcemy aby został on zamieniony na html. W tym celu wykorzystamy zainstalowaną bibliotekę **Markdig**. Proszę zamienić fragment kodu HTML odpowiedzialnego za wyświetlanie tekstu na taki:

```
<div class="col-6 h-100">
  @((MarkupString)@Markdig.Markdown.ToHtml(_markdownCode))
</div>
```

23. Proszę pobrać plik gif który znajduje się pod [linkiem](#), i zapisać go w folderze images pod nazwą banana.gif



24. Aby sprawdzić czy nasz interpreter działa poprawnie proszę w textarea (pole tekstowe użytkownika) wpisać ten tekst:

```
---
__Blazor__
---

## Code
Inline `code`
Indented code

    // Some comments
    line 1 of code
    line 2 of code
## Images

![Banana](images/banana.gif)
```

25. Tekst powinien zostać poprawnie zamieniony na html i wyświetlany z prawej strony tak jak by to zrobiła przeglądarka. Efekt końcowy został przedstawiony na rysunku poniżej.

## Interpreter

```
---
  Blazor
---

## Code
Inline `code`
Indented code

    // Some comments
    line 1 of code
    line 2 of code
## Images

![Minion]
(https://phoneky.co.uk/thumbs/screensavers/down/misc/banana\_dk6i3asu.gif)
```

## Blazor

### Code

Inline `code` Indented code

```
// Some comments
line 1 of code
line 2 of code
```

### Images

