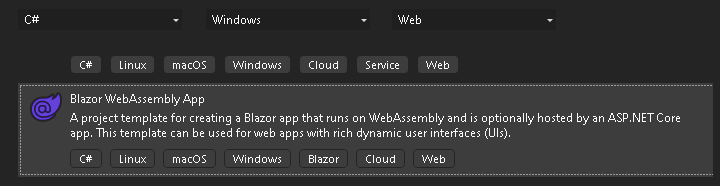
MVVM webalkalmazás készítése

# Blazor webassambly

### Projekt készítés

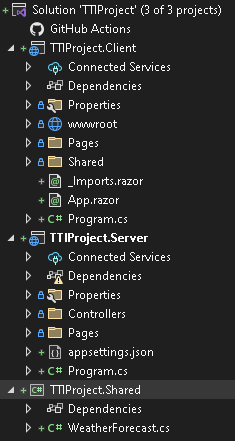
Készítsen projektet TTIProject néven:

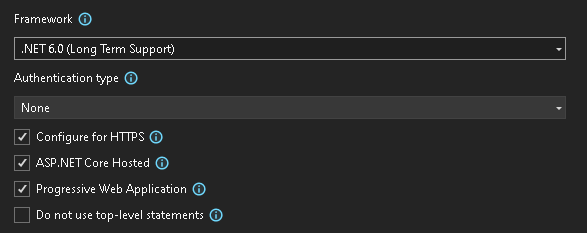
1.



2,



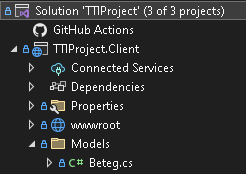
3.



Elkészül a frontent és a backend project is:

### Fronted készítése

1. A frotnend (cliens) projektben models mappába elhelyezzük a már használt Beteg.cs állományt.



1. A ViewModels mappán belül a BaseClass almappába helyezzük el a szükséges ViewModel kiegészítéseket.

Vegyük észre, hogy a ViewModelBase osztály kiegészül a IsBusy property-vel.

public class ViewModelBase : INotifyPropertyChanged

{

private bool isBusy = false;

public bool IsBusy

{

get => isBusy;

set

{

SetValue(ref isBusy, value);

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = null)

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

protected void SetValue<T>(ref T backingFiled, T value, [CallerMemberName] string propertyName = null)

{

if (EqualityComparer<T>.Default.Equals(backingFiled, value)) return;

backingFiled = value;

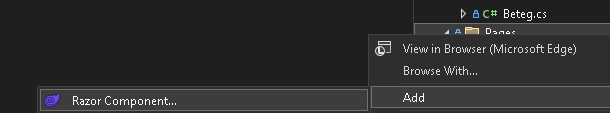
OnPropertyChanged(propertyName);

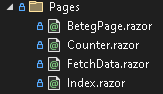
}

}

### Web oldal egy lapjának elkészítése

1. Készítsünk egy BetegPage razor oldalt a Pages mappába:





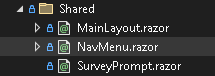
1. Adjuk meg melyik url-en lesz elérhető az oldal és módosítsuk a lap címét:

@page "/beteg"

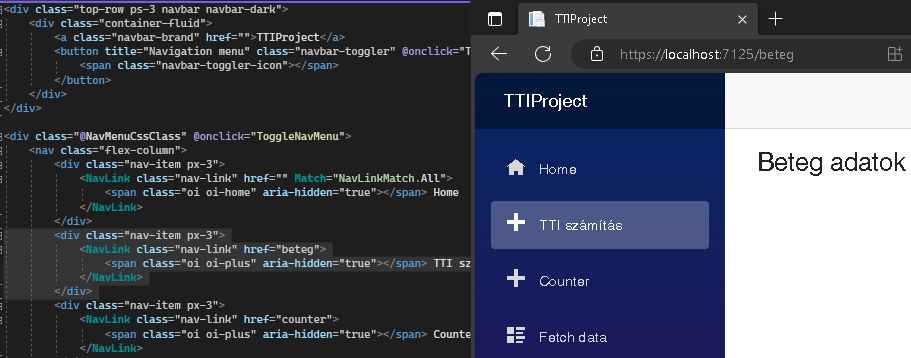
<h3>Beteg adatok</h3>

1. Menüpont hozzáadása a projekthez. Teszteljük a projekt működését.

A Shared menüpont NavMenu.razor fájlába dolgozunk!



A menüpont elhelyezése után a tesztelhetjük a weboldalt!



## Blazor MVVM

### Beteg osztály

1. A Beteg osztályt a ViewModelBase osztálytól örököltetjük

public class Beteg : ViewModelBase

{

1. Pontosítjuk a Beteg osztályba aza adatokat, property-ket

### Beteg oldal

1. A Beteg osztály lesz a ViewModel
2. Megírjuk az MVVM kódokat az oldalon (a kód bemásolható).

@page "/beteg"

<h3>Beteg adatok</h3>

@code {

private Beteg ViewModel = new Beteg();

protected override async Task OnInitializedAsync()

{

ViewModel.PropertyChanged += async (sender, e) =>

{

await InvokeAsync(() =>

{

StateHasChanged();

});

};

await base.OnInitializedAsync();

}

async void OnPropertyChangedHandler(object sender, PropertyChangedEventArgs e)

{

await InvokeAsync(() =>

{

StateHasChanged();

});

}

public void Dispose()

{

ViewModel.PropertyChanged -= OnPropertyChangedHandler;

}

1. Megírjuk a Form-ot. A form adatai a ViewModel-ből jönnek.

page "/beteg"

@using System.ComponentModel;

@using TTIProject.Model;

**<h3>Beteg adatok</h3>**

**<EditForm Model="@ViewModel" OnValidSubmit="@Compute">**

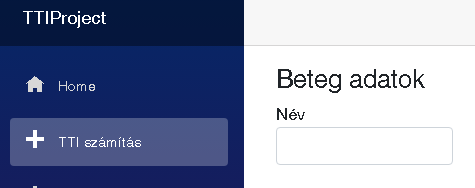
**<div class="col-4">**

…

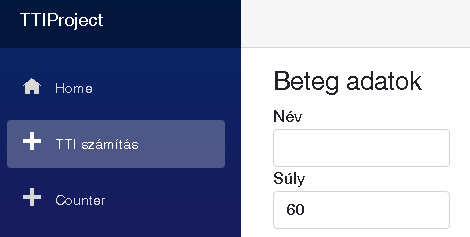
@code {

**private Beteg ViewModel = new Beteg();**

1. Az első bekért a adat a beteg neve lesz



1. A következő bekért adat a suly
2. A formon InputText controlban bekért adatok szöveges típusúak. Módosítjuk a Beteg osztályt



1. A következő bekért adat a magasság
2. Az eredményt a TTI meghatározás gombbal számoljuk
3. Az eredményt valid adatok esetén a Compute metódussal számoljuk majd
4. Eredmény számítás (javíthatjuk a gomb megjelenését)

