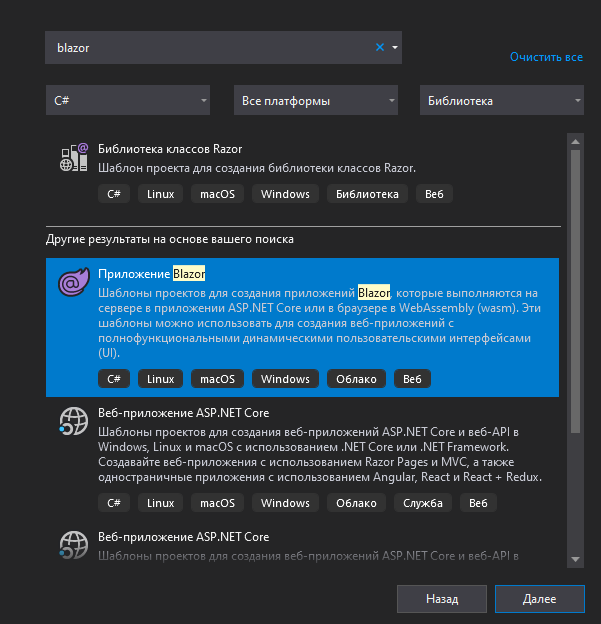
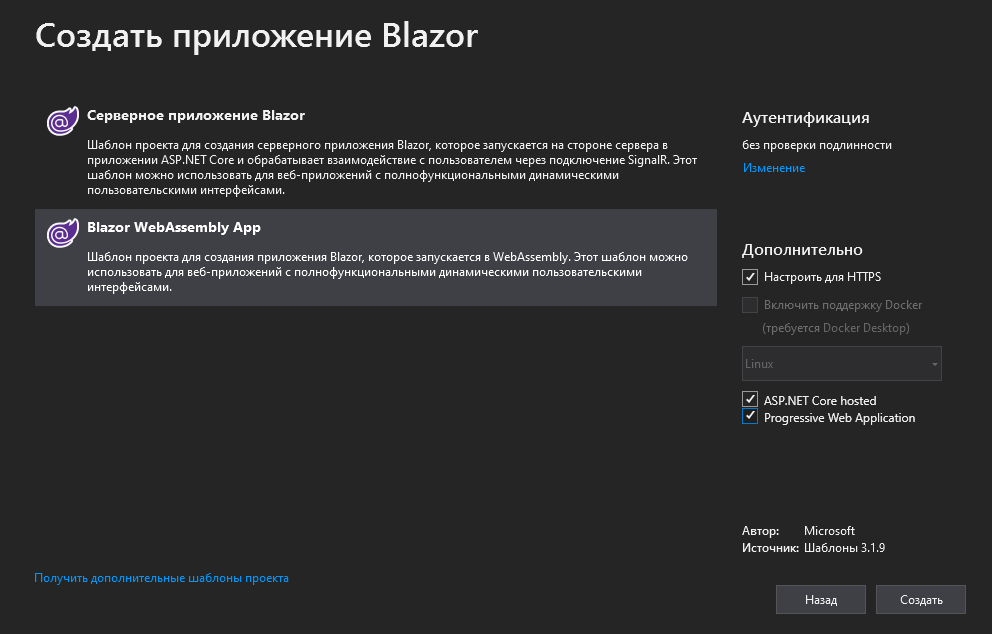
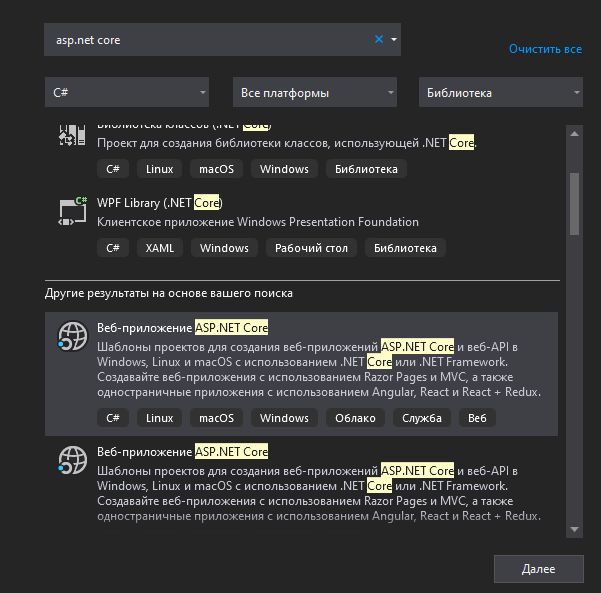
1. Создать приложение Blazor



* 1. Blazor WebAssembly App
  2. ASP.NET Core hosted
  3. Progressive Web Application



1. Добавить проект Веб-приложение ASP.NET Core



2.1) Тип шаблона – Api

1. В Api установить пакет Microsoft.AspNetCore.SignalR
2. В Api добавить класс

public class NotificationHub:Hub

{

public override Task OnConnectedAsync()

{

return base.OnConnectedAsync();

}

}

1. правим файл Startup.cs проекта Api

а) дописать в ConfigureServices

services.AddCors(

o =>

{

o.AddPolicy("CorsPolicy", policy =>

{

policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();

});

});

services.AddSignalR();

б) дописать в Configure

app.UseCors("CorsPolicy");

app.UseEndpoints(endpoints =>

{

endpoints.MapHub<NotificationHub>("/notificationhub");

endpoints.MapControllers();

});

1. В проект Shared добавить класс и enum

public class NotifiMessage

{

public string Title { get; set; }

public DateTime? Time { get; set; } = DateTime.Now;

public NotifiType Type { get; set; } = NotifiType.**none**;

public bool IsPrivate { get; set; } = true;

public string FromUserId { get; set; }

public string ToUserId { get; set; }

}

public enum NotifiType

{

**none**,

**Debug**,

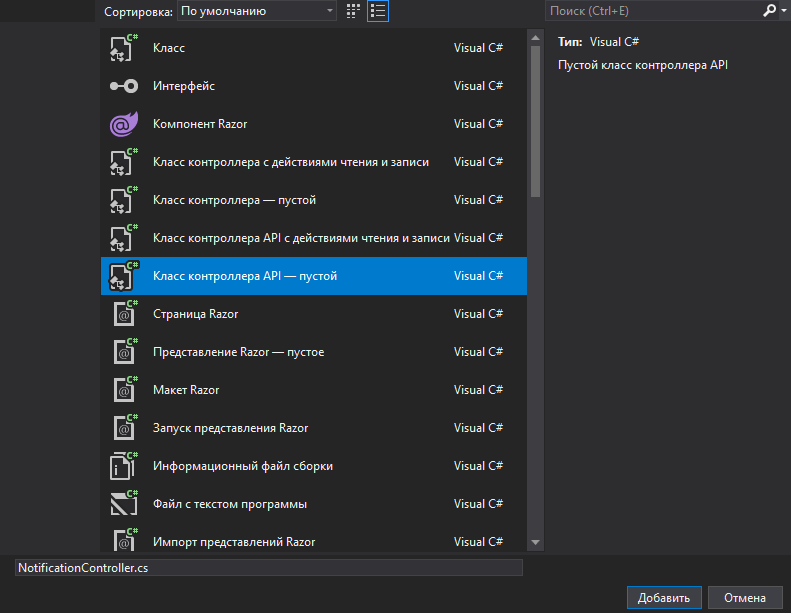
**Info**,

**Warning**,

**Error**

}

1. В проект с Api добавить пустой контроллер Api



Листинг

[Route("api/[controller]")]

[ApiController]

public class NotificationsController : ControllerBase

{

private readonly IHubContext<NotificationHub> \_HubContext;

private readonly ILogger<NotificationsController> \_Logger;

public NotificationsController(IHubContext<NotificationHub> hubContext, ILogger<NotificationsController> Logger)

{

\_HubContext = hubContext;

\_Logger = Logger;

}

[HttpGet]

public string Get()

{

return "It work";

}

/// <summary>

/// Получает информационное сообщение отправленное в теле запроса

/// </summary>

/// <returns></returns>

[HttpPost("SendTitle")]

public async Task<IActionResult> SendTitle()

{

using var reader = new StreamReader(Request.Body);

var json = await reader.ReadToEndAsync();

var inbound = JsonConvert.DeserializeObject<string>(json);

await \_HubContext.Clients.All.SendAsync("notification", new NotifiMessage { Title = inbound });

return Ok("Notification has been sent succesfully!");

}

/// <summary>

/// Получает информационное сообщение отправленное в строке

/// </summary>

/// <param name="Title">сообщение</param>

/// <returns></returns>

[HttpPost]

public async Task<IActionResult> Post([FromQuery] string Title)

{

await \_HubContext.Clients.All.SendAsync("notification", new NotifiMessage { Title = Title });

return Ok("Notification has been sent succesfully!");

}

/// <summary>

/// Получает информационное сообщение и рассылает его пользователям

/// </summary>

/// <param name="message">сообщение</param>

/// <returns></returns>

[HttpPost("SendMessage")]

public async Task<IActionResult> SendMessage(NotifiMessage message)

{

try

{

var id = message.FromUserId;

message.FromUserId = string.Empty;

if (message.IsPrivate)

await \_HubContext.Clients.Client(id).SendAsync("notification", message);

else

await \_HubContext.Clients.All.SendAsync("notification", message);

return Ok("Notification has been sent succesfully!");

}

catch (Exception e)

{

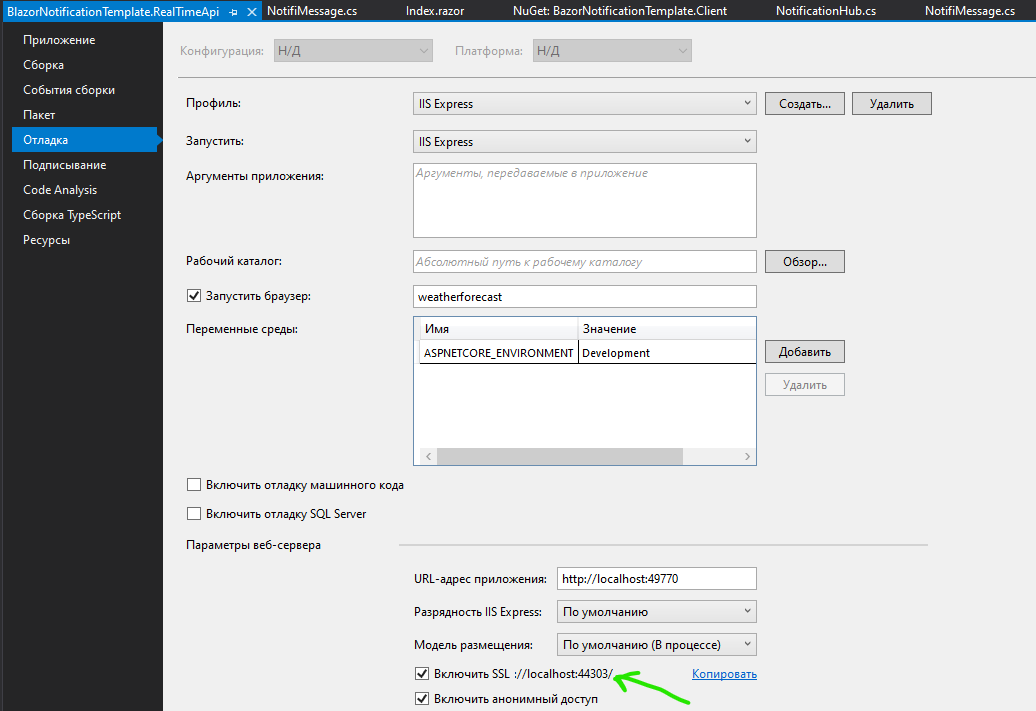
return BadRequest($"Error to send notification: {e.Message}");

}

}

}

1. В Client установить пакет - Microsoft.AspNetCore.SignalR.Client **не путать с Microsoft.AspNet.SignalR.Client**
2. В свойствах проекта Api взять адрес подключения - у меня это <https://localhost:44303/>



1. В клиент добавить листинг в файл index.razor

@page "/"

@using Microsoft.Extensions.Logging

@using BlazorNotificationTemplate.Shared

@using Microsoft.AspNetCore.SignalR.Client

@inject HttpClient client

@inject ILogger<Index> Logger

<h3>Connection Status: @connectionStatus</h3>

<div class="row">

<div class="col-8">

@foreach (var item in notifications)

{

<div class="row">

<h4>@item.Type</h4>

<h2>@item.Time</h2>

<h3>@item.Title</h3>

</div>

}

</div>

</div>

@code{

#region Notification

string url = "https://localhost:44303/notificationhub";

List<NotifiMessage> notifications = new List<NotifiMessage>();

HubConnection \_Connection = null;

private bool \_IsConnected = false;

bool IsConnected { get; set; }

string connectionStatus = "Closed";

protected override async Task OnInitializedAsync()

{

await ConnectToSrver();

}

private async Task ConnectToSrver()

{

\_Connection = new HubConnectionBuilder()

.WithUrl(url)

.Build();

await \_Connection.StartAsync();

IsConnected = true;

connectionStatus = "Connected";

\_Connection.Closed += async (s) =>

{

IsConnected = false;

connectionStatus = "Disconnected";

await \_Connection.StartAsync();

IsConnected = true;

connectionStatus = "Connected";

};

\_Connection.On<NotifiMessage>("notification", m =>

{

notifications.Add(m);

StateHasChanged();

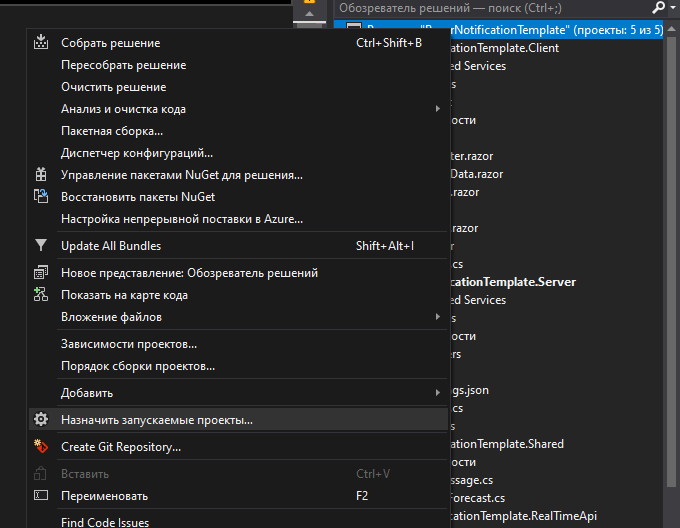
});

}

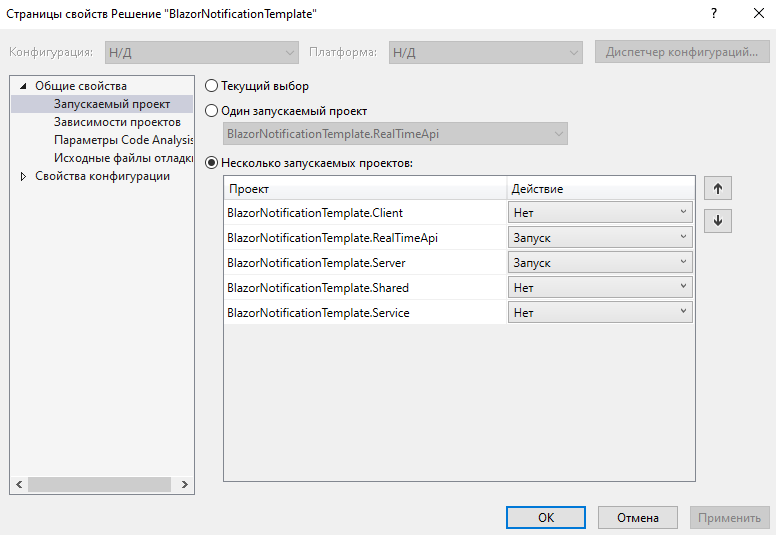
#endregion

}

1. Правой кнопкой на решении – назначить запускаемые проекты

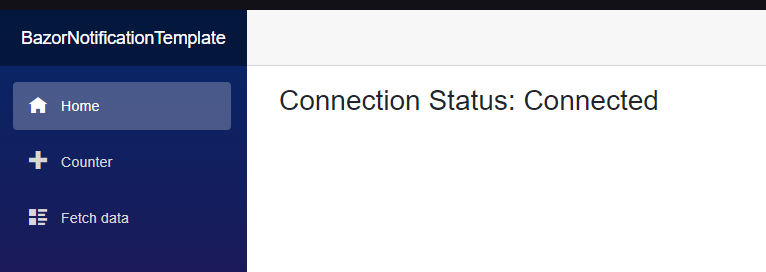


Устанавливаем запуск api И сервера приложения



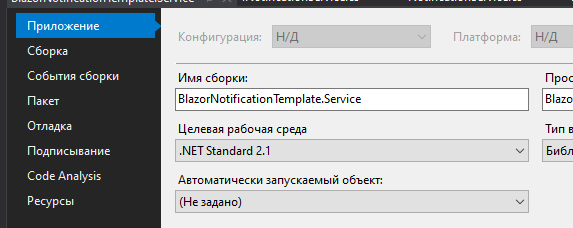
1. Запускаем проект для теста соединения

Видим что статус подключения через пару секунд меняется на “Connected”



1. Создать сервис для уведомлений

13.1) Добавить проект – библиотека Net.Standard 2.1

Если создастся 2.0 – проверить в свойствах проекта - 

13.2) в проект добавить интерфейс

using System.Threading.Tasks;

using BlazorNotificationTemplate.Shared;

namespace BlazorNotificationTemplate.Service

{

public interface INotificationService

{

public Task<bool> SendNotificationAsync(NotifiMessage message);

public Task<bool> SendNotificationAsync(string Title);

}

}

13.3) установить в проект пакет - System.Net.Http.Json

13.4) добавить в проект реализацию интерфейса

using System;

using System.Net.Http;

using System.Net.Http.Json;

using System.Threading.Tasks;

using BlazorNotificationTemplate.Shared;

using Microsoft.Extensions.Logging;

namespace BlazorNotificationTemplate.Service

{

public class NotificationService : INotificationService

{

private readonly ILogger<NotificationService> \_Logger;

public NotificationService(ILogger<NotificationService> Logger)

{

\_Logger = Logger;

}

#region Implementation of INotificationService

public async Task<bool> SendNotificationAsync(NotifiMessage message)

{

try

{

using var response = await new HttpClient().PostAsJsonAsync("https://localhost:44303/api/notifications/SendMessage", message);

return response.IsSuccessStatusCode;

}

catch (Exception e)

{

\_Logger.LogError("Ошибка отправки сообщения на сервер: {error}", e.Message);

return false;

}

}

public async Task<bool> SendNotificationAsync(string Title)

{

try

{

using var response = await new HttpClient().PostAsJsonAsync("https://localhost:44303/api/notifications/SendTitle", Title);

return response.IsSuccessStatusCode;

}

catch (Exception e)

{

\_Logger.LogError("Ошибка отправки сообщения на сервер: {error}", e.Message);

return false;

}

}

#endregion

}

}

Внимательно проверьте https адресс, хост должен быть тот-же что в Inex.razor клиента

1. В проект Server добавить ссылку на проект Сервисов

14.1) подключить реализацию в классе Startup.cs

services.AddSingleton<INotificationService, NotificationService>();

1. В проекте Server добавить контроллер

[ApiController]

[Route("[controller]")]

public class NotificationTestController : ControllerBase

{

private readonly ILogger<NotificationTestController> logger;

private readonly INotificationService \_Notification;

public NotificationTestController(ILogger<NotificationTestController> logger, INotificationService notification)

{

this.logger = logger;

\_Notification = notification;

}

[HttpGet("GetSomeData/{UserId}")]

public async Task<IActionResult> GetSomeData(string UserId)

{

for (var **i** = 0; **i** < 10; **i**++)

{

await \_Notification.SendNotificationAsync(new NotifiMessage {Title = $"Step {**i**}", FromUserId = UserId, Type = NotifiType.**Info**});

//long operation

await Task.Delay(1000);

}

return Ok();

}

}

1. Отредактировать Index.razor проекта Client

16.1) Добавить разметку

<button class="btn btn-info" @onclick="StartTest">Start Test</button>

<div class="row">

<div class="col-8">

@foreach (var item in notifications)

{

<div class="row card-header">

<span><b>@item.Type</b><p>@item.Time : @item.Title</p></span>

</div>

}

</div>

</div>

16.2) Добавить код вызова контроллера, передав ему ключ вызывающего пользователя

async Task StartTest()

{

var result = await client.GetAsync($"NotificationTest/GetSomeData/{\_Connection.ConnectionId}");

if (result.IsSuccessStatusCode)

{

}

}

Запускаем проект, статус меняется на “Connected”, жмём старт и видим процесс обработки идущий на сервере

