**Лабораторная работа №3**

**Цель работы:** реализовать Web приложение для манипулирования данными использующее Web API (из лр2) используя фреймворк blazor.

**Теория:**

1. **Blazor (кратко)**

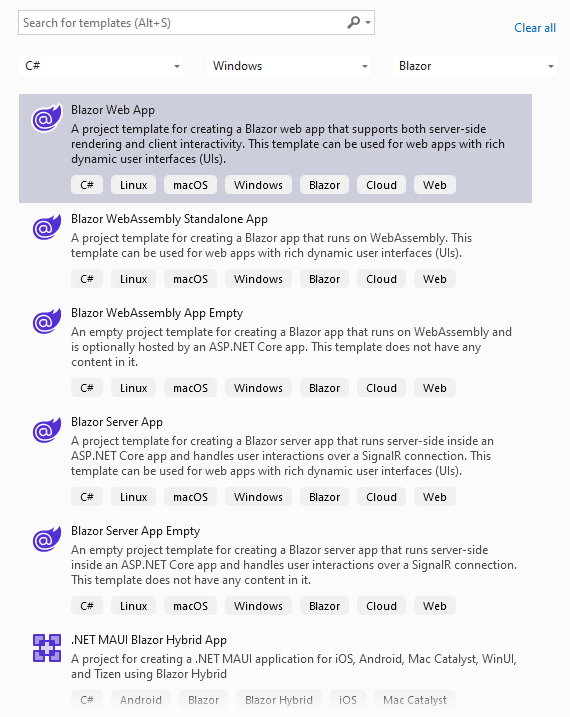
Blazor представляет UI-фреймворк для создания интерактивных приложений, которые могут работать как на стороне сервера, так и на стороне клиента, на платформе .NET. В своем развитии фреймворк Blazor испытал большое влияние современных фреймворков для создания клиентских приложений - Angular, React, VueJS. В частности, это проявляется в роли компонентов при построении пользовательского интерфейса. В то же время и на стороне клиента, и на стороне сервера при определении кода в качестве языка программирования применяется C#, вместо JavaScript. А для описания визуального интерфейса используются стандартные HTML и CSS.

Blazor предоставляет разработчикам следующие преимущества:

1. Написание кода веб-приложений с помощью C# вместо JavaScript
2. Использование возможностей экосистемы .NET, в частности, библиотек .NET при создании приложений, безопасности и производительности платформы .NET
3. Клиентская и серверная части приложения могут использовать общую логику
4. Использование Visual Studio в качестве инструмента для разработки, который имеет встроенные шаблоны для упрощения создания приложения
5. **Типы проектов blazor**

Функционально на текущий момент Blazor подразделяется на несколько подсистем:

1. Blazor Server: позволяет создавать серверные приложения и поддерживается ASP.NET
2. Blazor WebAssembly: позволяет создавать одностраничные интерактивные приложения клиентской стороны, которые запускаются в браузере пользователя и работают с помощью технологии WebAssembly\*
3. Blazor Hybrid: позволяет создавать десктопные и мобильные приложения поверх технологии .NET MAUI

****

Рекомендую использовать Blazor Web App.

1. **WebAssembly**

WebAssembly (Wasm) — это новый открытый формат байт-кода, исполняемого современными браузерами. Сейчас он поддерживается во всех браузерах.

Wasm позволяет переносить код, написанный на языках C, C++, C#, Rust и других, для использования в браузерной среде с производительностью, близкой к нативной. **Важно понимать, максимальная производительность WebAssembly равна максимальной производительности JavaScript**, но написать оптимизированный по памяти код иногда проще на WebAssembly.

Язык WebAssembly предназначен для дополнения и работы вместе с JavaScript, что позволяет веб-разработчикам использовать преимущества обоих языков:

* JavaScript — это язык высокого уровня, гибкий и достаточно выразительный для написания веб-приложений. У него много преимуществ: он динамически типизирован, не требует компиляции и обладает огромной экосистемой, которая предоставляет мощные фреймворки, библиотеки и другие инструменты.
* WebAssembly — это низкоуровневый язык, похожий на ассемблер, с компактным двоичным форматом, который работает с почти естественной производительностью и предоставляет низкоуровневые модели памяти для таких языков как C++ и Rust, чтобы компилировать их для работы в интернете. Обратите внимание, что у WebAssembly на будущее есть высокоуровневая цель по поддержке языков со сборщиками мусора в модели памяти.

1. **Razor page**

Приложение в Razor Pages организовано в виде страниц Razor. Каждая страница представляет пользовательский интерфейс и связанную с ним логику, причем определение пользовательского интерфейса от связанной с ним логики, что облегчает разработку приложения и его тестирование. Для определения пользовательского интерфейса применяется движок Razor, а для описания связанной со страницей логики - C#.

Частично, этот тип проекта будет повторять общую структуру проектов ASP.NET Core. Главным же отличием данного проекта от других типов проектов ASP.NET Core будет наличие папки Pages. Эта папка содержит все страницы Razor, которые есть в проекте по умолчанию. Вкратце организация проекта:

* Properties: содержит настройки проекта, в частности, файл launchSettings.json, который определяет настройки запуска проекта
* wwwroot: эта папка предназначен для хранения статических файлов. По умолчанию здесь уже есть ряд скриптов javascript и файлов css, в частности, файлы фреймворка bootstrap и библиотек валидации.
* Pages: содержит все страницы Razor. По умолчанию здесь имеются следующие файлы:
  + \_Layout.cshtml: мастер-страница, в которую вставляются страницы Razor
  + \_ViewStart.cshtml: задает мастер-страницу
  + \_ViewImports.cshtml: определяет директивы Razor, которые добавляются на каждую страницу Razor
  + \_ValidationScriptsPartial.cshtml: частичное представление, которое подключает js-скрипты валидации на стороне клиента
  + Index.cshtml, Error.cshtml и Privacy.cshtml: собственно страницы Razor, которые определяют визуальную часть страницы и логику обработки запроса.
* appsettings.Development.json: хранит конфигурацию приложения для стадии разработки
* appsettings.json: хранит конфигурацию приложения
* helloapp.csproj: главный файл проекта, который определяет его конфигурацию
* Program.cs: файл, который определяет класс Program, с которого начинается работа приложения. То есть это входная точка в приложение.

Каждая страница Razor представляет файл с расширением .cshtml и содержит смесь кода html и конструкций C#. С каждой страницей Razor связан файл отделенного кода логики на C#. Например, с файлом Index.cshtml, который определяет визуальную часть с помощью синтаксиса Razor (HTML + C#), связан файл Index.cshtml.cs, который определяет логику страницы или ее поведение с помощью кода C#.

Аналогично обстоит дело с другими страницами - Error.cshtml и Privacy.cshtml, с которыми связаны соответственно файлы кода Error.cshtml.cs и Privacy.cshtml.cs

1. **Настройка CORS (C**ross-O rigin **R**esource **S**haring**)**

При обращении к Web API вы можете столкнуться с ошибкой (ошибка может отображаться в консоли в средствах разработчика в браузере):

*Access to fetch at 'http://localhost:82/api/Repository' from origin 'http://localhost:5152' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled*

Это происходит потому что система безопасности браузера предотвращает запросы веб-страницы к другому домену, отличному от того, который обслуживает веб-страницу. Это ограничение называется политика одного источника. Эта политика предотвращает чтение вредоносным сайтом конфиденциальных данных с другого сайта. Иногда вы должны разрешить другим сайтам выполнять запросы к приложению независимо от источника.

Для решения в Program.cs (проекта **web api**) нужно разрешить права:

*// Add services to the container.*

*builder.Services.AddCors(options =>*

*{*

*options.AddPolicy("AllowBlazorApp",*

*builder =>*

*{*

*builder.AllowAnyOrigin()*

*.AllowAnyMethod()*

*.AllowAnyHeader();*

*});*

*});*

*app.UseCors("AllowBlazorApp");*

**Источники информации:**

1. <https://metanit.com/sharp/blazor/1.2.php> - Введение в blazor
2. <https://learn.microsoft.com/ru-ru/aspnet/core/blazor/?view=aspnetcore-8.0> - blazor
3. <https://ru.wikipedia.org/wiki/WebAssembly> - WASM
4. <https://habr.com/en/articles/475778/> - WASM
5. <https://habr.com/en/companies/vk/articles/775852/> - WASM
6. <https://metanit.com/sharp/razorpages/> - Razor pages
7. <https://learn.microsoft.com/ru-ru/aspnet/core/security/cors?view=aspnetcore-9.0> - CORS

**Задание:**

1. Создать в существующем решении (ЛР2) проект Web приложения используя шаблон Blazor Web App.
2. Настроить запуск двух проектов в visual studio.
3. Реализовать интерфейс приложения, позволяющий выполнять все операции с данными, предоставляющие API, а именно: просмотр/добавление/редактирование/удаление данных.
4. Протестировать функциональность приложения.
5. Для отображения коллекции данных использовать функции, предоставляемые razor pages, такие как for/foreach.
6. Предусмотреть информативное для пользователя сообщение о возникшей ошибке.

**Контрольные вопросы**

1. Что такое CORS. Как реализовали доступ к данным.
2. Продемонстрировать ход выполнения от нажатия клавиши до отображения данных.
3. Продемонстрировать использование возможностей razor page.
4. Продемонстрировать как устроена страница razor.
5. Продемонстрировать обработку исключений.