ОБ ОДНОЙ ТЕХНОЛОГИИ ПОВТОРНОГО ИСПОЛЬЗОВАНИЯ ИСХОДНОГО КОДА В КЛИЕНТ-СЕРВЕРНЫХ ПРИЛОЖЕНИЯХ

Семёнов М.Г.,
доцент кафедры прикладного и системного программирования ВГУ имени П.М. Машерова, канд. физ.-мат. наук, г. Витебск, Республика Беларусь

Ключевые слова. Клиент-серверная архитектура, Blazor , C#, ASP.NET Core.
Keywords. Client-server architecture, Blazor, C#, ASP.NET Core.

В настоящее время клиент-серверная архитектура является наиболее распространенным подходом в разработке многопользовательских информационных систем и веб-приложений. Приложение в таком случае разделяется на клиентскую и серверную части. При этом каждая из частей может разрабатываться отдельно от другой и, более того, может использовать различные языки программирования.

Наиболее распространенные технологии для программирования бизнес-логики на клиентской стороне (такие как ReactJS, Angular, Vue и другие) основаны на использовании языка программирования JavaScript. С появлением HTML 5 позиции JavaScript, как основного языка программирования для клиентской разработки только усилились. Вместе с тем, на серверной стороне применяется довольно большое количество различных технологий с использованием таких языков, как Java, C#, Python, PHP и других.

Ряд задач бизнес-логики серверной и клиентской частей бывают связаны, а иногда и полностью дублируют друг друга. Ярким примером таких задач является проверка и верификация данных. Однако, в связи с различием технологий, применяемых на клиентской и серверной частях приложения, как правило, приходиться выполнять двойную работу. В связи с этим, возникает вопрос о разработке общей базы исходного кода для обеих частей. Одним из существующих вариантов решения данной проблемы является применение на серверной части языка программирования JavaScript совместно с технологией NodeJS. Вместе с тем, успешных примеров применения языков серверной части для программирования клиентской бизнес-логики на текущий момент нет.

В 2020 году компанией Microsoft была выпущена технология Blazor, которая позволяет использовать код, написанный на языке C#, для программирования клиентской части приложения. *Целью* данной работы является анализ возможностей технологии Blazor для разработки современных клиент-серверных приложений.

Материал и методы. Материалом для исследования послужили современные многопользовательские информационные системы и веб-приложения. При проведении исследований применялись общепризнанные методы научного познания.

Результаты и их обсуждение. Ядро Blazor представляет строго типизированную среду на основе веб-браузера [1], в которой можно создавать веб-интерфейсы с использованием языка C# и платформы .NET Core вместо JavaScript. Работа Blazor основана на относительно новой технологии под названием WebAssembly. WebAssembly открыла возможность компилировать код, который не обязательно является JavaScript, в модули с байт-кодом низкого уровня (WASM), которые веб-браузеры могут выполнять напрямую без необходимости синтаксического анализа исходного файла. Примечателен тот факт, что на текущий момент WebAssembly поддерживается всеми основными веб-браузерами последних версий. В связи с этим, применение технологии Blazor не требует установки дополнительного программного обеспечения на стороне пользователей.

Для выполнения модулей .NET технология Blazor использует версию среды выполнения .NET, скомпилированную в WASM. В свою очередь, Blazor располагается поверх основной среды выполнения и реализует механизм Razor, используемый в качестве точки входа в код .NET, который может обрабатываться браузером. Это позволяет применять

большинство подходов, которые обычно используются при разработке на платформе .NET (например, импортировать и ссылаться на дополнительные сборки .NET).

Приложения Blazor основаны на компонентах. Компонент в Blazor представляет собой часть пользовательского интерфейса, такие как страницы, диалог или форму ввода данных. С точки зрения кода, компоненты – это классы C #, встроенные в сборки .NET, которые:

1. определяют логику рендеринга пользовательского интерфейса;
2. описывают ообработку пользовательских событий;
3. можно вложить друг в друга и использовать повторно;
4. могут распространяться в виде библиотек классов или пакетов NuGet.

Существует две модели хостинга приложений с использованием Blazor: Blazor WebAssembly и Blazor Server.

Первый подход базируется на использовании WebAssembly. При первом обращении к веб-странице, пользователем скачиваются файлы WASM среды исполнения, самого приложения и зависимостей. Стоит отметить, что размеры данных файлов небольшие и соизмеримы бандлами популярных JavaScript технологий.

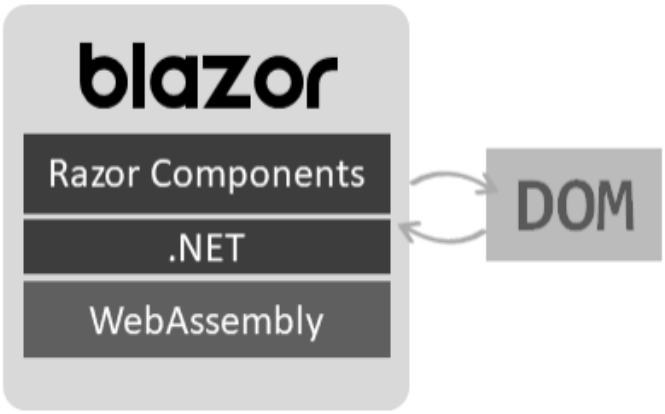

Рисунок 1. Схема работы Blazor WebAssembly

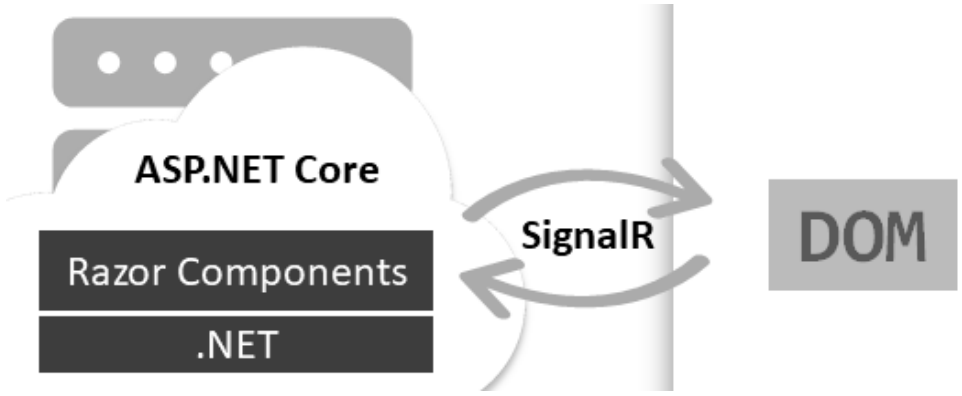

Рисунок 2. Схема работы Blazor Server

Blazor Server в свою очередь логика приложения исполняется на стороне сервера, а обновление интерфейса и обработка событий осуществляется с помощью SignalR соединения. Стоит отметить, что, несмотря на схожесть данного подхода с классическим рендерингом на стороне сервера, существует ряд существенных различий в этих подходах (смотри, например, [2]).

Заключение. Благодаря технологии WebAssembly, Blazor представляет собой возможную альтернативу популярным JavaScript технологиям Angular, React и Vue. Данный подход открывает возможность применения преимуществ существующей экосистемы .NET, а также написания общих библиотек бизнес-логики для клиентской и серверной части, что позволит значительно снизить затраты на разработку программного обеспечения, основанного на клиент-серверном подходе.

1. Esposito, D. Never Mind JavaScript, Here's Blazor / D. Esposito // MSDN Magazine – V. 33, N. 9 – 2018. – Режим доступа: https://docs.microsoft.com/en-us/archive/msdn-magazine/2018/september/cutting-edge-never-mind-javascript-here%E2%80%99s-blazor

2. ASP.NET Core Blazor hosting models [Электронный ресурс] Режим доступа: https://docs.microsoft.com/en-us/aspnet/core/blazor/hosting-models?view=aspnetcore-5.0

ПСЕВДОДОПОЛНЕНИЯ В РЕШЁТКЕ ФОРМАЦИЙ

Столяренко А.Ю.,
студентка 4-го курса ВГУ имени П.М. Машерова, г. Витебск, Республика Беларусь
Научный руководитель – ***Мехович А.П.,*** *канд. физ.-мат. наук*

Ключевые слова. Псевдодополнение, решётка формаций, формация конечных групп, подформация, простой делитель порядка группы.

Keywords. Pseudocomplement, the lattice of formations, formation of finite groups, subformation, prime divisor of the order of the group.

Изучение решёток формаций конечных групп было начато в 1986 г. А. Н. Скибой, где была установлена модулярность решётки всех формаций [1]. Этот результат получил развитие в различных направлениях. В частности, в работе [2] описаны стоуновы решётки n-кратно насыщенных формаций и псевдодополнения в такой решётке.

Целью данной работы является описание псевдодополнения в решётке всех подформаций формации $\mathfrak{F}$.

Материал и методы. В работе используются терминология и методы доказательства абстрактной теории групп и их классов, в частности теории формаций конечных групп.

Результаты и их обсуждение. Все рассматриваемые нами группы конечны. Напомним, что частично упорядоченное множество, в котором для любых двух элементов существует точная нижняя и точная верхняя грани, называется решёткой. Пусть L – решётка с нулём. Тогда элемент $a^* \in L$ называется *псевдодополнением* элемента $a \in L$, если из $a \wedge a^* = 0$ и $a \wedge x = 0$ следует, что $x \leq a^*$ [3].

Формацией называется класс групп, замкнутый относительно гомоморфных образов и конечных подпрямых произведений, т. е. если выполняются следующие условия:

1) $G \in \mathfrak{F}$ и $N \lhd G$, то $G / N \in \mathfrak{F}$;

2) $G / N_1 \in \mathfrak{F}$ и $G / N_2 \in \mathfrak{F}$, то $G / N_1 \cap N_2 \in \mathfrak{F}$.

В дальнейшем для всякого непустого множества простых чисел π через π', $G_{\pi'}$ и (1) обозначают соответственно дополнение к π во множестве всех простых чисел, класс всех π' групп, формацию всех единичных групп. Символ $\pi(G)$ обозначает множество всех раз-