

Рубленко Маргарита Віталіївна

Здобувачка вищої освіти факультету інформаційних технологій
Державний торговельно-економічний університет, Україна

Науковий керівник: Юрченко Юрій Юрійович

Старший викладач, кафедра комп'ютерних наук та інформаційних систем
Факультет інформаційних технологій
Державний торговельно-економічний університет, Україна

BLAZOR – ЯК ПРАЦЮЄ ФРЕЙМВОРК

Blazor є сучасним веб-фреймворком, розробленим компанією Microsoft, який дозволяє створювати інтерактивні веб-застосунки з використанням мови програмування C# замість традиційного JavaScript. Його поява стала важливим етапом розвитку платформи ASP.NET Core, оскільки він забезпечує можливість об'єднання клієнтської та серверної частини застосунку в межах єдиної технологічної екосистеми [1]. Основна ідея Blazor полягає у використанні вже знайомих .NET-технологій для побудови сучасних веб-інтерфейсів, що значно спрощує процес розробки та зменшує необхідність вивчення додаткових мов і фреймворків [4].

Принцип роботи Blazor базується на компонентному підході, який передбачає створення інтерфейсу у вигляді незалежних, повторно використовуваних компонентів. Кожен компонент поєднує HTML-розмітку та логіку на C#, використовуючи спеціальний Razor-синтаксис. Завдяки цьому розробник може описувати як зовнішній вигляд, так і поведінку елементів інтерфейсу в одному файлі. Важливою особливістю є підтримка автоматичного оновлення інтерфейсу при зміні даних, що реалізується через механізм прив'язки даних (data binding) [2].

Blazor підтримує два основних режими виконання — Blazor Server та Blazor WebAssembly, які відрізняються місцем обробки логіки застосунку. У випадку Blazor Server вся обробка відбувається на сервері, а браузер виконує роль відображення інтерфейсу. Взаємодія між клієнтом і сервером реалізується за допомогою технології SignalR, що забезпечує постійне двостороннє з'єднання. Кожна дія користувача передається на сервер, обробляється і повертається у вигляді DOM-структури сторінки [1].

На відміну від цього, Blazor WebAssembly дозволяє виконувати код безпосередньо у браузері користувача. У цьому випадку C#-код компілюється у формат WebAssembly — спеціальний двійковий формат, який підтримується сучасними браузерами і забезпечує високу швидкість виконання [5]. Завдяки

цьому застосунок може працювати автономно після завантаження, зменшуючи навантаження на сервер і забезпечуючи швидку реакцію інтерфейсу. Однак одним із недоліків цього підходу є значний розмір початкового завантаження, оскільки разом із застосунком завантажуються середовище виконання .NET [3, 5].

Архітектура Blazor включає кілька ключових механізмів, які забезпечують його ефективність і гнучкість. Одним із таких механізмів є система маршрутизації, яка дозволяє створювати односторінкові застосунки з переходами між сторінками без перезавантаження. Іншим важливим елементом є Dependency Injection — вбудований механізм впровадження залежностей, який дозволяє керувати життєвим циклом об'єктів і спрощує тестування та масштабування застосунку. Також Blazor використовує диференційний підхід до оновлення DOM, що означає, що при зміні стану компоненту оновлюється лише необхідна частина інтерфейсу, а не вся сторінка [2, 4]. Життєвий цикл компонентів у Blazor дозволяє контролювати поведінку застосунку на різних етапах. Компоненти проходять через етапи ініціалізації, отримання параметрів, рендерингу та оновлення. Розробник може перевизначати відповідні методи життєвого циклу для виконання додаткової логіки, наприклад, завантаження даних із сервера або обробки подій користувача [1].

Серед основних переваг Blazor варто виділити можливість використання однієї мови програмування для всієї логіки застосунку, що знижує складність розробки та підтримки. Крім того, інтеграція з екосистемою .NET дозволяє використовувати широкий спектр бібліотек та інструментів, що вже існують. Це особливо важливо для корпоративних застосунків, де часто потрібна складна бізнес-логіка та інтеграція з іншими системами [4]. Водночас слід враховувати і певні обмеження, такі як відносно новий статус технології, що означає меншу кількість готових рішень і бібліотек у порівнянні з популярними JavaScript-фреймворками [3].

Таким чином, Blazor є потужним і перспективним фреймворком, який змінює підхід до створення веб-застосунків, дозволяючи використовувати можливості платформи .NET у повному обсязі як на сервері, так і на стороні клієнта. Його використання сприяє підвищенню продуктивності розробки, зменшенню кількості технологій у стеку та спрощенню підтримки програмного забезпечення, що робить його привабливим вибором для сучасних розробників.

Список використаних джерел:

1. Microsoft. Blazor documentation [Електронний ресурс]. – Режим доступу:

- <https://learn.microsoft.com/aspnet/core/blazor>
2. Freeman A. *Pro ASP.NET Core 6*. – 9th ed. – New York: Apress, 2022. – 1000 p.
 3. Price M. *C# 10 and .NET 6 – Modern Cross-Platform Development*. – Birmingham: Packt Publishing, 2022. – 820 p.
 4. Esposito D. *Modern Web Development with ASP.NET Core 3*. – Redmond: Microsoft Press, 2020. – 320 p.
 5. WebAssembly.org. WebAssembly Concepts [Электронный ресурс]. – Режим доступа: <https://webassembly.org>