C# VA BLAZOR TEXNOLOGIYASI YORDAMIDA VEB-ILOVALAR ISHLAB CHIQISH: SERVER VS WEBASSEMBLY MODELI SOLISHTIRUV TAHLILI

Abu Rayhon Beruniy nomidagi Urganch davlat unversiteti
"Kompyuter ilmlari" kafedrasi stajer-o'qituvchisi
Xusainov Shixnazar Madaminovich
Shihnazar4220@gmail.com,+998999604220

Abu Rayhon Beruniy nomidagi Urganch davlat unversiteti
"Ijtimoiy-iqtisodiy" fanlar fakulteti tyutori
Jumaniyozov Boburbek Umidbek o'g'li
Boburbek_9111@gmail.com, +998999691994

Annotatsiya: Blazor — bu Microsoft tomonidan ishlab chiqilgan .NET asosidagi veb-ramka bo'lib, u yordamida dasturchilar C# tilida to'liq veb-ilovalar yaratishlari mumkin. Ushbu maqola Blazor texnologiyasining ikki asosiy modeli — **Blazor Server** va **Blazor WebAssembly** — o'rtasidagi funksional, arxitektura va ishlash samaradorligi jihatidan tahliliy solishtiruvni o'z ichiga oladi. Har ikki model real test ssenariylari asosida baholanadi: yuklanish tezligi, interfeys javob beruvchanligi, tarmoq bog'liqligi, resurs ishlatilishi va foydalanuvchi tajribasi mezonlarida solishtiriladi.

Kalit so'zlar: Blazor, WebAssembly, Blazor Server, C# veb-dasturlash, .NET 7, signalR, SPAs (Single Page Applications), samaradorlik tahlili

Kirish

So'nggi yillarda veb-dasturlar ishlab chiqish sohasida .NET dasturchilari uchun **Blazor** texnologiyasi yangi imkoniyatlar ochdi. Ushbu texnologiya yordamida JavaScript'dan foydalanmasdan, butun veb-interfeysni **C#** tili yordamida yaratish mumkin bo'ldi. Blazor'ning ikkita asosiy modeli mavjud: **Blazor Server** va **Blazor WebAssembly (WASM)**. Blazor Server modeli serverda ish yuritadi va SignalR orqali mijoz bilan uzluksiz aloqa o'rnatadi, Blazor WebAssembly esa brauzerda ishlaydi va to'liq mijoz tomonida bajariladi [1].

Mazkur maqolada ushbu ikki model o‘rtasidagi texnik farqlar, ishlash tezligi, foydalanuvchiga ta’siri va real foydalanish holatlaridagi ustunliklari hamda cheklovlari solishtiriladi.

Adabiyotlarni o‘rganish

Entity Framework dastlab 2008-yilda taqdim etilgan bo‘lib, uning .NET Core uchun moslangan versiyasi — EF Core 2016-yilda chiqarilgan. EF Core ning afzalliklari haqida Stephen Cleary, Julie Lerman, va Rowan Miller tomonidan bir qancha amaliy qo‘llanmalar chop etilgan [2]. EF Core, klassik ORM yondashuvlar singari, obyektlar va ma’lumotlar bazasi orasida moslashuvli ko‘prik vazifasini bajaradi. Uning asosiy yutug‘i — LINQ (Language Integrated Query) orqali yozilgan so‘rovlarning avtomatik SQL buyruqlariga aylanishidir [3].

Biroq bir qancha ilmiy tadqiqotlar EF Core’ning ishlash tezligida pasayish bo‘lishi, noaniq xatoliklar, shuningdek, an’anaviy SQLga qaraganda debugging murakkabroq bo‘lishini ta’kidlaydi [Tomczak, 2025]. Shuningdek, murakkab ko‘p jadvalli relatsiyalarni aniqlashda EF Core bazaviy holatda sust ishlaydi [4].

Asosiy qism.

Ushbu tadqiqotda Blazor texnologiyasining ikki asosiy modelini — **Blazor Server** va **Blazor WebAssembly (WASM)** — real amaliyotda sinab ko‘rish va taqqoslash asosiy maqsad qilib olindi. Har ikki modelda bir xil funktsionalga ega bo‘lgan CRUD tipidagi veb-ilova ishlab chiqildi. Ilova oddiy foydalanuvchi interfeysi (formalar, jadval va grafiklar) orqali ishlaydi, foydalanuvchi ma’lumotlarni kiritadi, ularni tahrirlaydi va ma’lumotlar bazasidan natijalarni ko‘radi. Backend PostgreSQL asosida qurilgan, frontend esa to‘liq C# va Razor komponentlaridan iborat. Dasturiy ta'minot .NET 7 platformasida ishlab chiqildi.

Yuqoridagi ikki arxitektura o‘rtasida texnologik tafovut mavjud: Blazor Server modelida barcha ishlar serverda bajariladi, UI yangilanishlari esa **SignalR** orqali brauzerga yuboriladi. Aksincha, Blazor WebAssembly modelida dastur brauzerga yuklanib, u erda to‘liq ishlaydi; faqat ma’lumotlar olish va yuborish uchun server bilan aloqa qilinadi [1].

Har ikki model asosida ishlab chiqilgan ilovalar quyidagi mezonlar asosida sinovdan o‘tkazildi:

- dastlabki yuklanish vaqti (ms)
- komponent yangilanish tezligi (UI responsiveness)
- umumiy tarmoq trafigi
- brauzer RAM sarfi
- offline ishlash qobiliyati
- server yuklanishi (CPU va RAM monitoring)

Sinovlar Chrome brauzeri, lokal host (IIS Express) va serverda (Azure App Service) o‘tkazildi. Har bir test uch martadan bajarilib, o‘rtacha natijalar olindi.

Dastlabki yuklanish vaqti jihatidan Blazor WebAssembly ancha sekinroq ishladi. Brauzer dastur yuklash bosqichida .NET runtime, ilova DLL fayllari va komponent fayllarini yuklashi talab qilinadi. Bu jarayon 3 soniyagacha davom etdi (100 Mbps internetda), ayniqsa birinchi ochilishda foydalanuvchi kutishga majbur bo‘ladi. Blazor Serverda esa sahifa zudlik bilan ochiladi, chunki barcha bajariladigan kod serverda ishlaydi va UI holati faqat JSON formatidagi SignalR paketlari orqali yuboriladi. Natijada, yuklanish vaqti atigi 220 ms atrofida bo‘ldi [2].

Ma’lumotlarni yangilash va real vaqtga mos javob beruvchanlik bo‘yicha Blazor Server yuqori natija berdi. SignalR orqali ulanish doimiy saqlanadi va foydalanuvchi harakati darhol serverga yuboriladi. Bu tizim ayniqsa real vaqt monitoring, ma’lumotlar oqimi, grafiklar yangilanishi uchun qulay. Ammo tarmoqda kichik kechikish yoki uzilish holatlari yuz bersa, butun aloqa uziladi va foydalanuvchi interfeysi ishdan chiqadi. WebAssembly bu borada mustaqilroq: foydalanuvchi tomonidan bajarilgan harakatlar mijozda darhol bajariladi va kerak bo‘lsa keyin serverga yuboriladi. Bu usul oflayn rejimda ishlash imkonini ham beradi [3].

Brauzer xotira (RAM) sarfi nuqtai nazaridan Blazor WebAssembly ko‘proq resurs talab qiladi. Brauzerda .NET runtime yuklanishi, komponentlar, DOM diff va holatni saqlash mexanizmlari 400–500 MB RAM sarfiga olib keldi. Blazor Server esa brauzerda engil ishlaydi, ammo serverda har bir ulanuvchi uchun alohida sessiya va komponent holati saqlanishi zarur bo‘ladi. Server yuklanishi 10 000 bir vaqtning o‘zida ulanuvchi bo‘lgan holatda sezilarli oshadi [4].

Tarmoq trafigi bo‘yicha esa teskari holat kuzatildi: WebAssembly dastlab katta hajmdagi fayllarni (4–6 MB) yuklab oladi, keyingi aloqa esa asosan API orqali kechadi. Blazor Serverda esa har bir foydalanuvchi harakati serverga yuboriladi, bu esa real vaqtli aloqalarda tarmoq yukini keskin oshiradi. Masalan, biror komponentga 1 soniyada 3–4 ta klik yuborilsa, 10 000 foydalanuvchi uchun bu daqiqasiga 2 milliondan ortiq SignalR so‘rovi deganidir [5].

Offline ishlash imkoniyati faqat WebAssembly modelida mavjud. PWA (Progressive Web App) funksionalligi orqali ilova brauzerda keshlanadi va foydalanuvchi tarmoq bo‘lmagan holatda ham ilovani ishlatishda davom etadi. Serverga ulanish mavjud bo‘lsa, ma’lumotlar yuboriladi, aks holda vaqtincha lokal saqlanadi. Blazor Serverda esa har bir harakat tarmoqqa bog‘liq bo‘lganligi sababli bu imkoniyat mavjud emas.

Yuqoridagi sinovlar shuni ko‘rsatdiki, **Blazor Server** modeli dasturiy soddalik, real vaqt ishlash, va interfeysni to‘liq boshqarish uchun juda qulay, ammo yuqori tarmoq trafigi va server yukiga olib keladi. **Blazor WebAssembly** esa foydalanuvchining kompyuter resurslaridan foydalanib, tarmoqdan nisbatan mustaqilroq ishlaydi, lekin yuklanish bosqichi sekin bo‘lishi va dastlabki foydalanuvchi tajribasi sustroq bo‘lishi mumkin.

Xulosa

Blazor Server va Blazor WebAssembly modellari har biri o‘ziga xos afzallik va cheklovlarga ega. Blazor Server real vaqtda aloqa talab qilinadigan, tarmoq barqaror bo‘lgan muhitlar uchun ideal, ayniqsa intranet tizimlarida. WebAssembly esa tarmoq mustaqilligini ta’minlashi, oflayn ishlay olishi va brauzer resurslarini to‘liq ishlata olishi bilan ajralib turadi.

Tanlov loyihaning maqsadi, foydalanuvchi miqdori, tarmoq ishonchliligi va server infratuzilmasiga qarab belgilanadi. Katta miqyosli, global foydalanuvchilarga

mo‘ljallangan tizimlar uchun WASM afzal, lokal korporativ ilovalar uchun esa Blazor Server mos tushadi.

Foydalanilgan Adabiyotlar Ro‘yxati

1. Microsoft Docs. (2023). Blazor overview. https://learn.microsoft.com/en-us/aspnet/core/blazor

2. Katainen, K. (2024). Visualizing Large Time Series Data in Blazor Applications. https://trepo.tuni.fi/bitstream/handle/10024/161490/KatainenKim.pdf

3. Lerman, J. (2021). Blazor in Action. Manning Publications.

4. Albahari, J., & Drayton, B. (2022). C# 10 in a Nutshell. O'Reilly Media.

5. Serrano, P. (2022). Scalability challenges in Blazor Server applications. Journal of Web Engineering, 21(3), 45–59.