

DOI: <https://doi.org/10.36910/6775-2524-0560-2025-59-33>

УДК 004.415.3

Pekh Petro, PhD

<https://orcid.org/0000-0002-6327-3319>

Yanchar Oleksandr, bachelor

Lutsk National Technical University, Lutsk, Ukraine

## FEATURES OF MODERN TECHNOLOGY C# BLAZOR SERVER FOR WEB APPLICATIONS DEVELOPMENT

**Pekh P., Yanchar O. Features of modern technology for C# Blazor Server for web applications development. creating web applications using C# BLAZOR SERVER.** The article examines the features of using the C# Blazor Server platform using the example of developing a web service for job search. Probably, no one disputes the statement that a convenient and fast web service for job search is equally needed, on the one hand, by those who are looking for a job, and on the other hand, by those who offer this job. Therefore, the development of simple and fast web services of this direction is and will be an urgent task in the future. What is the main difficulty here? In our opinion, this is the optimal choice of the platform by means of which the web service should be created. Based on the fact that to solve the tasks that arise in the process of job search, one cannot do without developing and using databases containing information about both clients and employers, the advantage is on the side of platforms that provide such functionality. C# Blazor Server belongs to such platforms. As for the main advantage of the C# Blazor Server platform, the researchers focus on the fact that all the code is executed on the server, and the client only receives updates via SignalR. This allows the client to avoid downloading a large amount of .NET runtime. At the same time, the requirements for the browser are reduced, since the service works even on medium-power devices. In addition, the service is quickly started without delays in loading and compiling WebAssembly.

**Keywords:** C# ASP .NET platform, C# Blazor Server technology, SignalR. WebAssembly, web service

**Пех П.А., Янчар О.Р. Особливості сучасної технології C# Blazor Server створення веб-додатків.** В статті досліджуються особливості використання платформи C# Blazor Server на прикладі розробки веб-сервісу для пошуку роботи. Мабуть, ні в кого не викликає заперечення твердження, що зручний та швидкодіючий веб-сервіс для пошуку роботи однаково потрібен, з одного боку, тим хто роботу шукає, а, з другого боку тим, хто цю роботу пропонує. Тож розробка простих та швидкодіючих веб-сервісів подібного спрямування є і буде в майбутньому актуальною задачею. В чому тут основна складність? На нашу думку, це - оптимальний вибір платформи, засобами якої веб-сервіс має створюватися. Виходячи з того, що для вирішення задач, які виникають у процесі пошуку роботи, не обійтися без розробки і використання баз даних, що містять інформацію, як про клієнтів, так і про роботодавців, перевага на боці платформ, які такий функціонал забезпечують. C# Blazor Server належить саме до таких платформ. Щодо основної переваги платформи C# Blazor Server, то увага дослідників акцентується на тому, що увесь код виконується на сервері, а клієнт лише отримує оновлення через SignalR. Це дозволяє клієнту не завантажувати великий обсяг .NET-рантайму. При цьому зменшуються вимоги до браузера, оскільки сервіс працює навіть на середніх за потужністю пристроях. Крім того, забезпечується швидкий старт сервісу без затримки на завантаження та компіляцію WebAssembly.

**Ключові слова:** платформа C# ASP .NET, технологія C# Blazor Server, SignalR. WebAssembly, веб-сервіс

**Problem statement.** The need for a fast and convenient job search remains relevant for many users, as does the need for effective recruitment for employers. Most modern platforms are difficult to use, have high resource requirements, or are financially inaccessible to small businesses.

Given this, there is a need to create an optimized web service that will be simple, accessible, and will work stably even on medium-power devices.. One of the effective solutions is to use C# Blazor Server, which allows you to implement interactive functionality with minimal load on the client side.

**The purpose of the work is** to study the C# Blazor Server platform in the process of developing a web service for job search, which provides convenient and effective interaction between job seekers and employers.

The main functions of the web service should include:

- creating a profile and resume by users;
- searching for vacancies with the ability to apply filters;
- sending feedback on vacancies;
- creation and editing of vacancies by employers.

**The novelty of the work** lies in the use of Blazor Server technology to develop a web service for searching for jobs without using JavaScript on the client side. This approach allows you to implement all business logic exclusively on the server, which significantly reduces the requirements for the client device and ensures stable operation even on low-power systems.

**The key features** of the implementation are:

- saving the user interface state on the server, which guarantees data integrity and stability of interaction;

- using the SignalR protocol for two-way communication between the client and the server in real time;
- updating the DOM on the client side without completely reloading the page, which improves performance and is convenient for users;
- building a system based on a modern architecture with a clear distribution of functions, which ensures easy scaling and the possibility of further expansion of functionality.

**The main part.** The web service developed in the work, the structure of which is shown in Figures 1 and 2, is implemented using a three-tier architecture that provides modularity, extensibility and ease of system support:

1. Client layer – built using Razor components that implement interfaces for registration, creating resumes and vacancies. The components provide interactivity and dynamic interaction with the user without completely reloading the pages.

2. Server layer – includes C# classes for processing business logic and a service layer that is responsible for interaction with the database. Communication between the client and the server is carried out via SignalR, which allows updating the interface in real time.

3. Data storage layer – represented by a SQL Server database, the models of which are implemented using Entity Framework Core, which provides convenient object-relational mapping (ORM) and supports data migration.

**Additional technologies used in the project.** A number of modern technologies were used to implement the functionality of the web service, ensuring scalability, security, speed, and ease of development. Table 1 lists the main components of the system and the corresponding technologies used for their implementation.

Table 1 – Technologies used in the project

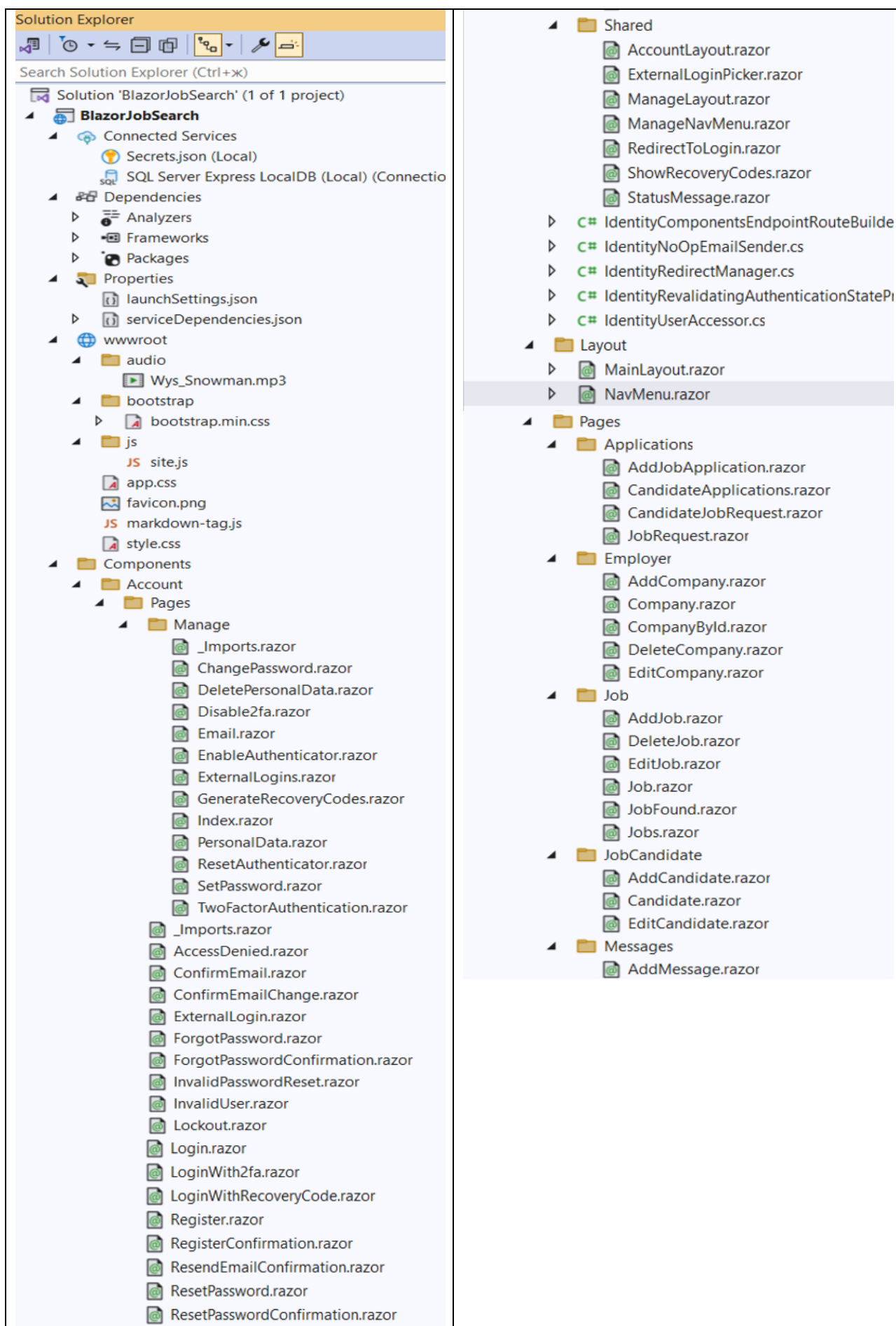
Component	Technology used
UI Components	Blazor Server, CSS
ORM	Entity Framework Core + LINQ
Authentication	ASP.NET Core Identity
Communication	SignalR
Database	SQL Server
Configuration Storage	appsettings.json
Security	ASP.NET Middleware

**System security.** ASP.NET Identity is used to protect data, which provides:

- user registration with identity confirmation (email confirmation);
- authorization according to access roles;
- access restriction to confidential information;
- protection against major web threats, such as CSRF and XSS, through built-in Razor Pages mechanisms and middleware.

**Using SignalR for real-time.** A feature of Blazor Server is the presence of a built-in SignalR connection, which provides instant page updates when the status changes. For example:

- when a candidate applies for a vacancy, the employer instantly sees a new message in his personal account;
- when the application status changes, the candidate also sees the update without a reboot.



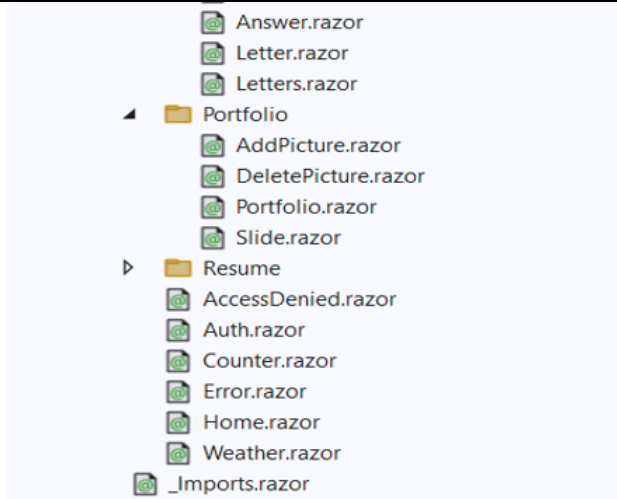
	 <p>The screenshot displays a file explorer window with a light blue background. It shows a directory structure with the following files and folders:</p> <ul style="list-style-type: none"><li>Answer.razor</li><li>Letter.razor</li><li>Letters.razor</li><li>Portfolio (folder, expanded)<ul style="list-style-type: none"><li>AddPicture.razor</li><li>DeletePicture.razor</li><li>Portfolio.razor</li><li>Slide.razor</li></ul></li><li>Resume (folder, collapsed)<ul style="list-style-type: none"><li>AccessDenied.razor</li><li>Auth.razor</li><li>Counter.razor</li><li>Error.razor</li><li>Home.razor</li><li>Weather.razor</li></ul></li><li>_Imports.razor</li></ul>
--	--

Figure 1 – Web service structure (beginning))

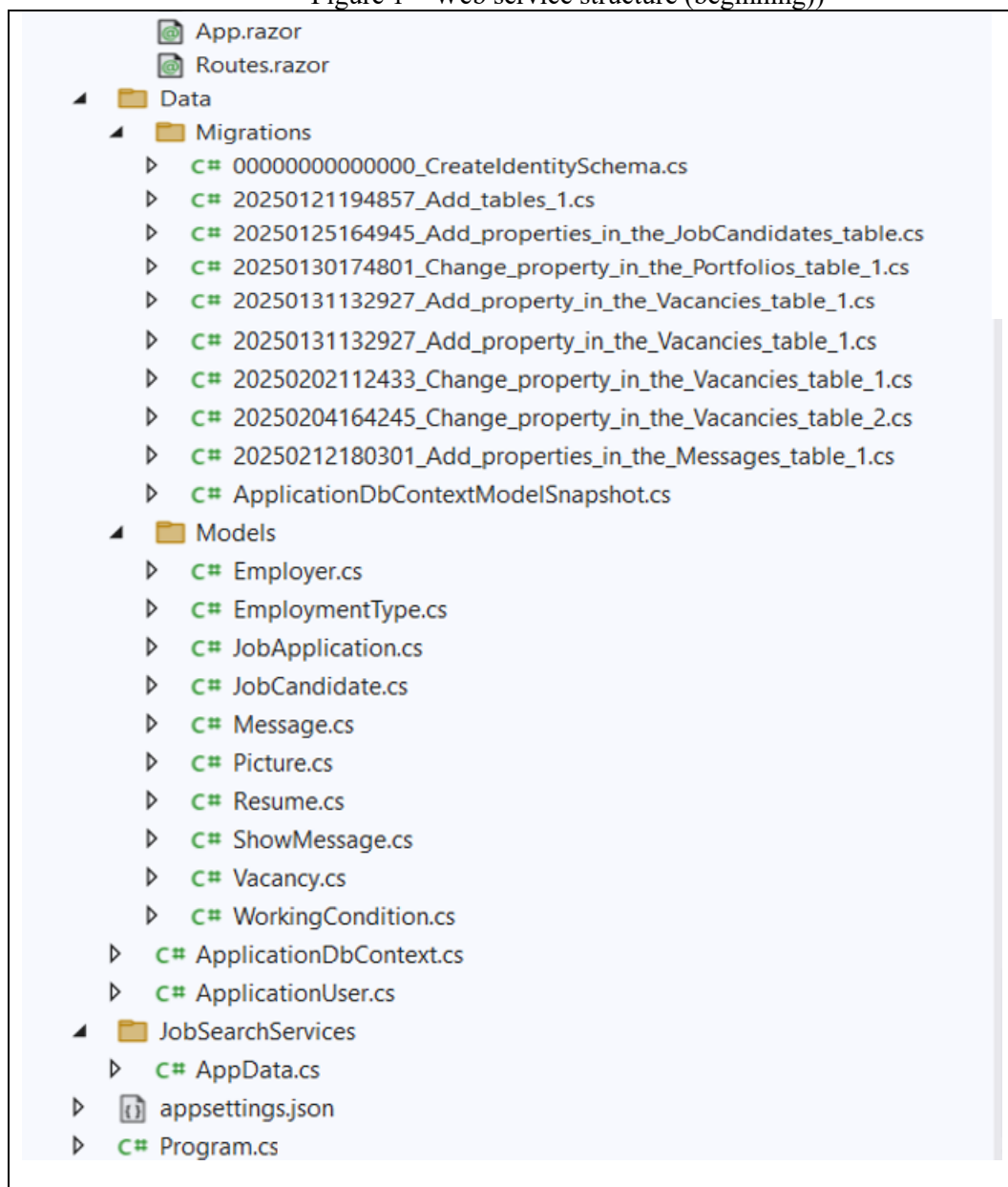


Figure 2 – Web service structure (end)

A generalized diagram of the system architecture and the interaction between its components is shown in Figure 3.



Figure 3 – Architecture of a job search web service

To implement the functionality of the web service, a database structure was developed that reflects key entities and their relationships. The main objects are system users (candidates and employers), resumes, vacancies, and job reviews. The main database entities with a list of key fields are given in Table 2.

Table 2 – Main database entities

Entity	Main fields
Users	Id, UserName, Email, PasswordHash, Role, CreatedAt

Resumes	Id, UserId, FullName, Experience, Skills, Education, CreatedAt
Vacancies	Id, EmployerId, Title, Description, Location, EmploymentType, Salary, CreatedAt
Applications	Id, ResumeId, VacancyId, Status, AppliedAt

Relationships between database entities:

Users – stores information about system users who can be candidates or employers (defined by the Role field).

Resumes are linked to users via the UserId field – each candidate can have one or more resumes.

Vacancies are linked to employers via the EmployerId field – each employer can create multiple vacancies.

Applications display candidate responses to vacancies by linking resumes (ResumeId) to vacancies (VacancyId). Thus, one candidate can apply for many vacancies, and a vacancy can receive many responses.

**Main system functionalities:**

- user registration and authorization with role differentiation (candidate, employer, administrator);
- viewing, searching and filtering vacancies by keywords, region, type of employment;
- creation and editing of resumes by users;
- creation, editing and publishing of vacancies by employers;
- platform administration (content moderation, user management).

**User interface implementation.** To provide a convenient and intuitive user interface, a set of Razor components was created that implement sections for candidates and employers. The main emphasis is on adaptability and dynamic content update without page reloading. Components are reused and interact through events (EventCallback) and state services.

**Communication with the server** is carried out through DI services that use HttpClient or directly call repository methods in Blazor Server. Figure 4 shows a fully functional registration page interface.

Figure 4 – Registration page interface

**Scalability and hosting.** To ensure scalability, Azure SignalR Service was used, which allows processing a large number of simultaneous connections. Redis Backplane is also used, which allows the system to scale horizontally in a clustered environment (Azure, Docker, Kubernetes).

The service is deployed on Azure App Service, using Azure SQL as the main database. Continuous integration and delivery (CI/CD) are implemented using GitHub Actions. Configuration parameters are stored in the appsettings.json file, and sensitive data is stored in Azure Key Vault.

### Conclusions.

The developed web service for job search showed the effectiveness of the approach with the execution of business logic on the server and updating the interface via SignalR. This ensures stable operation even on weak devices and a quick start without delays. The use of a three-tier architecture, Entity Framework Core, ASP.NET Identity, and scalable Azure services ensures security, convenience, and the ability to expand and improve the web service.

### References

1. ASP.NET Core Blazor. Microsoft Learn: Build skills that open doors in your career. URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-9.0> (access date: 07.05.2025).
2. Overview of ASP.NET Core SignalR. Microsoft Learn: Build skills that open doors in your career. URL: <https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-9.0> (access date: 07.05.2025).
3. Azure SignalR Service documentation. Microsoft Learn: Build skills that open doors in your career. URL: <https://learn.microsoft.com/en-us/azure/azure-signalr/> (access date: 07.05.2025).
4. Himschoot P. Blazor Revealed: Building Web Applications in .NET. Apress, 2019. 272 p.
5. ASP.NET Core Blazor authentication and authorization. Microsoft Learn: Build skills that open doors in your career. URL: <https://learn.microsoft.com/en-us/aspnet/core/blazor/security/?view=aspnetcore-9.0&tabs=visual-studio> (access date: 07.05.2025).