Abstract

This document is a synthesis of the researches done by Jonathan Racaud on the topic of dotnet Blazor

Techno Study

TOPIC = dotnet Blazor

Contents

[1. Introduction 0](#_Toc48739697)

[1.1. The techno 0](#_Toc48739698)

[1.2. Definitions 0](#_Toc48739699)

[1.3. Associated search terms 1](#_Toc48739700)

[2. The techno 1](#_Toc48739701)

[2.1. Added value 1](#_Toc48739702)

[2.2. Global trend 1](#_Toc48739703)

[2.3. Geographic distribution & trends per location 1](#_Toc48739704)

[2.4. Key applications 1](#_Toc48739705)

[2.5. Main challenges 1](#_Toc48739706)

[3. Alternatives 2](#_Toc48739707)

[3.1. Global alternative/competitor landscape 2](#_Toc48739708)

[3.2. Details about alternatives/competitors 2](#_Toc48739709)

[3.3. Features comparison 2](#_Toc48739710)

[3.4. Success stories 3](#_Toc48739711)

[4. The clients 3](#_Toc48739712)

[4.1. Client’s sectors 3](#_Toc48739713)

[4.2. Client’s size 3](#_Toc48739714)

[4.3. Example of clients 3](#_Toc48739715)

[5. Proof of concept 3](#_Toc48739716)

[5.1. Amaris Blazor Lab 3](#_Toc48739717)

[5.1.1. Conditions to meet to properly evaluate the benefits of the techno 3](#_Toc48739718)

[5.1.2. Methodology 4](#_Toc48739719)

[5.1.3. POC setup and development 4](#_Toc48739720)

[5.1.4. Assessment of the POC 5](#_Toc48739721)

[5.2. Blazor Rest Client 5](#_Toc48739722)

[5.2.1. Conditions to meet to properly evaluate the benefits of the techno 5](#_Toc48739723)

[5.2.2. Methodology 5](#_Toc48739724)

[5.2.3. POC setup and development 5](#_Toc48739725)

[5.2.4. Assessment of the POC 5](#_Toc48739726)

[6. Conclusion 5](#_Toc48739727)

[6.1. To continue further in the POC 5](#_Toc48739728)

[6.2. Alternative to try ? 5](#_Toc48739729)

[6.3. Comparison with other techno (subjective experience OK) 5](#_Toc48739730)

1. Introduction
   1. The techno

Blazor is a new technology created by Microsoft that allows writing C# application for the web without having to write JavaScript code for the front-end. It makes it possible to have most of the development made in mostly one language: C#. HTML and CSS are still used for the data display and styling.

There are two different models for a Blazor app respectively called **Blazor server-side** and **Blazor wasm**. Both models leverage the Web Assembly technology for the DOM manipulation.

*Blazor server-side:* Requires a constant connection between the server and the client (browser) as the server and the client communicate with Microsoft’s SignalR technology (based on websockets). All the heavy lifting of manipulating the DOM is made on the server side and only the difference between the old state and new state is sent through SignalR for the client to update the view. This model cannot be run offline.

*Blazor wasm:* The entirety of the application lives on the client side. Effectively a version of the .Net Core framework is downloaded by the client in order to run the application inside the browser thanks to Web Assembly. This allows the application to run offline. The main drawback of this model is that the amount of data to be downloaded is heavier than for a traditional web app.

It is possible to switch between the two models during the development.

The way we develop applications using Blazor is akin to the JavaScript framework Angular. So, component based.

It is possible to write JavaScript code to provide additional features that Blazor cannot provide at the moment (e.g.: canvas manipulation) because Web Assembly do not provide them yet.

Since May 19th, 2020 Blazor WebAssembly is officially released and considered ready for production by Microsoft [[Source](https://devblogs.microsoft.com/aspnet/blazor-webassembly-3-2-0-now-available/)].

There are also experiments to run Blazor on the desktop akin to Electron. One such experiment can be seen in this [blog post](https://blog.stevensanderson.com/2019/11/01/exploring-lighter-alternatives-to-electron-for-hosting-a-blazor-desktop-app/) by Steve Sanderson (part of the ASP .Net team at Microsoft).

* 1. Definitions

**.Net Core:** Microsoft’s open source and cross-platform framework to build console, web or desktop application. Several programming languages can be used, but usually it is C#.

**.Net Framework:** Legacy version of the .Net Framework that was only compatible with Windows. It is soon to be discontinued and .Net Core is the preferred framework to be used for any new C# projects.

**C#:** Microsoft’s programming language used with the Blazor technology.

**Razor:** Microsoft’s technology used in ASP .Net MVC which Blazor use for its components.

**SignalR**: Websocket based technology that allows real-time communication between the server side and client side of a Blazor application.

**Web Assembly (WASM):** Web technology that runs in the browser allowing to use different programming languages other than JavaScript for the front-end part of a web application. It provides a set of “Assembly” code for any compiled language to compile to. In the same way C or C++ is compiled into machine code.

**Websocket:** Web technology that brings the socket mechanism (used in the TCP/UDP protocols) for communication over the network between separated programs into the browsers.

* 1. Associated search terms

Razor, SignalR, WASM, Blazor server-side, Blazor wasm, Razor

1. The techno
   1. Added value

The added value of this technology is to be able to rely mostly on C# for developing a web application. This allows using a safer language for the development as the static type nature of C# allows to catch errors at compile time whereas JavaScript (without TypeScript) can only catch them at runtime.

Also, because Blazor runs on top of .Net Core, this means that one can use all the available libraries and framework compatible with the .Net Core framework (which is compatible for the most part with the previous .Net Framework).

[.NET is free](https://dotnet.microsoft.com/platform/free), and that includes Blazor. There are no fees or licensing costs, including for commercial use. - [Microsoft](https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor)

Also, if a team is familiar with traditional ASP .Net all their knowledge and skills will be easily transferable to Blazor, since it uses the same underlying technology (Razor).

* 1. Global trend

Blazor seems to gain a lot of attraction from the web development community. Microsoft links to this [awesome-blazor](https://github.com/AdrienTorris/awesome-blazor) github repository that provides an extensive list of resources around Blazor (libraries, books, podcast, udemy/skillshare courses, etc.).

People are also writing blog posts about the technology since its first introduction on plateform like [Medium.com](https://medium.com/search?q=Blazor).

Blazor also have a subreddit community that was created on August 12th, 2017 and has now 3.7K members.

* 1. Geographic distribution & trends per location

Looking at the [awesome-blazor](https://github.com/AdrienTorris/awesome-blazor) github repository, we can see that it exists community based content for Blazor in Russian, French, Italian and Portuguese.

* 1. Key applications

Even though Blazor can be used where JavaScript is used for the most part; games and application relying on the canvas might need more ‘hacks’ and workaround; one big key application that we can imagine and that was demoed by Microsoft, is enterprise web application such as intra-net web app.

The reason for that is that the requirement for the download size of the web application might be more laxed than for general consumer apps.

* 1. Main challenges

One of the main challenges for Blazor is the application binary size which is for the moment bigger than traditional approach because it ships the .Net Core Framework with it in order to run.

It also does not have access to the full capabilities of the DOM manipulation or browser features by itself, but that is more to do with the fact that Web Assembly does not cover all the scopes covered by JavaScript yet.

Another challenge will be its adoption for using the technology outside of the browser, so on the desktop or mobile space. It is possible to run Blazor with Electron and there are experiments to use a more lightweight alternative to Electron for the same goal [[Source](https://blog.stevensanderson.com/2019/11/01/exploring-lighter-alternatives-to-electron-for-hosting-a-blazor-desktop-app/)].

Also, with the announcement by Microsoft of [.Net MAUI](https://devblogs.microsoft.com/dotnet/introducing-net-multi-platform-app-ui/) (Multi-platform App UI) that is their effort to unify all of their GUI development tools, Blazor could be a target and so that could open the possibility to have Blazor used on mobile platforms.

1. Alternatives
   1. Global alternative/competitor landscape

Blazor really plays in the same space as all the other available web framework both on the front-end and back-end. Due to its nature it can be used only on the front-end for developing SPA app with Blazor wasm or run on both with Blazor server-side.

On the front-end, the current alternative to Blazor that are used in production are traditional JavaScript’s frameworks such as React, Angular or Vue.js.

For the backend it would be Symfony, Laravel or even ASP .Net MVC, but in the case of ASP .Net MVC they share a lot of similarities. The real difference being Blazor running on web assembly.

There are also web assembly frameworks for different languages such as Rust, C++ or Go.

[Vaadin](https://vaadin.com/) seems to also be a competitor that allows writing application for the web, mobile and desktop using Java only.

* 1. Details about alternatives/competitors

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Key features | Techno license | Strengths | CON | Pricing | Web site | Comments |
| Reactjs | Virtual DOM, JSX | MIT | Backed by Facebook |  | Free | <https://reactjs.org/> |  |
| Angular |  | MIT | Backed by Google |  | Free | <https://angular.io/> |  |
| Vue.js | Virtual DOM, Calculated properties | MIT | Lightweight |  | Free | <https://vuejs.org/> | Gains a lot of traction |
| Vaadin |  | Apache 2 or [cval-3.0](https://vaadin.com/license/cval-3.0) | Java only |  | Free – 870+€/months | <https://vaadin.com/> |  |

* 1. Features comparison

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Feature name/techno | Blazor | Reactjs | Angular | Vue.js | Vaadin | … | … |
| Mobile dev | Yes (experimental) | Yes | Yes | Yes | Yes |  |  |
| Web Assembly | Yes | No | No | No | No |  |  |
| JavaScript | JavaScript interop | Native | Native | Native | JavaScript interop |  |  |
| Desktop development | Yes (experimental) | Yes | Yes | Yes | Yes |  |  |

* 1. Success stories

1. The clients
   1. Client’s sectors

Consulting, training

* 1. Client’s size

International, small

* 1. Example of clients

[Cazton](https://www.cazton.com/consulting/web-development/blazor), [Redox Software](https://redox-software.co.uk/), [Nodexr](https://www.nodexr.net/),

1. Proof of concept
   1. Amaris Blazor Lab

A small project using Blazor server-side to manage projects and their contributors. This POC would show the capabilities of Blazor for developing “ERP” style projects

* + 1. Conditions to meet to properly evaluate the benefits of the techno

The following features must be implemented:

* Having users, projects, project materials and project categories. Material are images, videos or another document to accompany any project.
* Users should have two roles: Admin and Regular.
* Access to certain pages should be restricted based on the user’s role.
* Connection to a database.
* Having a front-end to display the different projects and users associated with them. As well as simple user dashboard.

Regular users should be able to:

* Login/Logout.
* Modify their profile.
* Join a project.
* Leave a project.
* Upload materials to a project. (To be implemented)

Admin are regular users who can also:

* Create regular users.
* Delete regular users.
* Create a project.
* Update a project (its description, name or category).
* Assign users to project.
* Remove users from project.
  + 1. Methodology
    2. POC setup and development

Prerequisites

You can find the source code for the project at this location: [location to be defined].

You will need to install the [.Net Core 3.1 SDK](https://dotnet.microsoft.com/download/) in order to run and build the project.

Make sure you have the ASP .Net and Web development workloads installed on your machine if using Visual Studio for development:

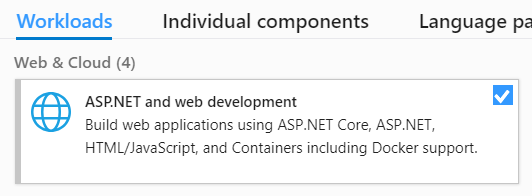


Figure . Visual Studio Installer - Workloads

You also need to have SQL Express installed on your development machine or provide the right connection string to an SQL Database inside the **appsettings.json** file at the root of the project.

Building the project

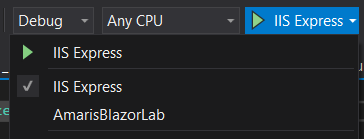
You can use Visual Studio 2019, Visual Studio Code or .Net Core CLI to build the project. If using Visual Studio, you need to make sure the ASP.Net and Web packages are installed.

**Visual Studio**

Before running the project, you need to run the migrations for creating and populating the database. So, using the Package Manager Console, run the following command:

$> Update-Database

When using Visual Studio, you can decide to run the project either with IIS Express or directly:



**Visual Studio Code/.Net Core CLI**

Before running the project, you need to run the migrations for creating and populating the database. For this you need to install the *dotnet ef* tool globally or locally:

$> dotnet tool install --global dotnet-ef

You can now run the following command to update the database:

$> dotnet-ef database update

With Visual Studio Code open a terminal to where the *AmarisBlazorLab.csproj* file is located, then you can run one of the following commands:

$> dotnet run: Will build and run the project.

$> dotnet watch run: Will build and watch the project, while also looking for file change to rebuild the project when changes to the code has been made.

P.S.: There is no debugger attached when using the above two commands.

The database should already be seeded with a default Admin user:

* Email: [admin@test.com](mailto:admin@test.com)
* Password: Admin#1

You can use this user to log in and test the features.

* + 1. Assessment of the POC
  1. Blazor Rest Client

Use Blazor wasm to create a Progressive Web App than consume a restful api. It exists several open api that could be called referenced here: [apilist.fun](https://apilist.fun). This POC would showcase the use of Blazor as a Single Page Application (SPA) framework. This would also show we can use it to build web app that work offline.

* + 1. Conditions to meet to properly evaluate the benefits of the techno
    2. Methodology
    3. POC setup and development
    4. Assessment of the POC

1. Conclusion
   1. To continue further in the POC

### **Amaris Blazor Lab**

* Implement the upload of files to the server and associate them with a specific project.
* Improve and fix some minor issues with the project update process.
* Improve the UI/UX and better manage the responsive approach.
* Add the possibility to connect using an OpenID Connect provider (Google, Microsoft, Apple, Github, etc).
  1. Alternative to try ?

### Vaadin, at first glance, seems to be very close in term of philosophy and usage compared to Blazor server-side and seems to also benefit from the Java ecosystem.

* 1. Comparison with other techno (subjective experience OK)

**Jonathan’s experience:** Using Blazor is a good experience, the access to the C# ecosystem really is a plus and makes it somewhat easy to develop what we want. The development speed is a little bit slower than with a JavaScript framework because of the compilation time. It is possible to have dotnet watch for file modification and update in real time the application during development, but it takes a little bit of effort to set it up that way. We are in a more traditional code -> compile -> run type of development cycle.

Even though it is a component-based framework, the CSS is not embedded inside the component like it is done with vue.js. It exists community based packages that can make it possible, but due to the nature of the framework and because it is in early stage the execution of this feature is a little bit finnicky and adds a lot of syntax that can clutter the code.

Blazor wasm is a little bit slow to load in my experience because of the download size of the application and the startup time so on my personal project using Blazor I preferred using Blazor server-side, but I don’t have the need of the web app to run offline.