# Vergleich ASP.NET Web App und Blazor

Im Rahmen unserer Diplomarbeit mussten wir uns entscheiden welches Webframework für unsere Web-App am besten war. Die Web-App sollte eine benutzerfreundliche Website werden die sowohl für Piolten als auch für Admins als Management Tool verwendet werden kann. Die Web-App ist in mehreren Seiten aufgebaut und umfasst Pilotenseiten und mehrere Admin/Management Seiten. Die Web-App benötigte deswegen eine Datenbankanbindung und zur Sicherheit muss noch eine Authentifizierung ermöglicht werden. Am Schluss musste die App noch für den Flugplatz Seitenstetten zur Verfügung gestellt werden.

Zur Auswahl standen

* ASP.NET Web App (Basis: MVC oder Razor Pages)
* Blazor (Server oder WebAssembly).

## Architekturen

#### Asp.Net Web App (MVC/Razor Pages):

Bei klassischen ASP.NET Webanwendungen wird HTML serverseitig generiert. Die Trennung erfolgt nach dem bekannten Model-View-Controlle. Für dynamische und interaktive Benutzeroberfläche ist meist ein zusätzlicher Einsatz von JavaScript oder Frontend-Frameworks wie Angular oder React notwendig. Der Client erhält dabei primär eine fertige HTML vom Server.

#### Blazor (Server oder WebAssembly):

Blazor ermöglicht es, Webanwendungen vollständig in C# zu entwickeln. Sowohl Frontend als auch Backendlogik können in derselben Sprache implementiert werden. Statt Controller und Views werden Komponenten verwendet, ähnlich wie in modernen JavaSript Frameworks. Änderungen in der UI werden direkt über Komponentenlogik umgesetzt, wodurch eine besser dynamische Website ermöglicht wird.

## Entwicklungsaufwand

#### Asp.Net Web App (MVC/Razor Pages):

Entwickler müssen zwischen C#, Razor und JavaScript wechseln. Das erhöht den Aufwand und die Komplexität. Weiters muss man für Razor, JavaScript und den Frameworks zusätzliches Wissen aufbringen was die Recherche und den Einstiegsaufwand erhöhen.

#### Blazor (Server oder WebAssembly):

Im Gegensatz zur ASP.NET WebApp wird in Blazor nur mit C# gearbeitet. Dadurch entfallen die Recherche zu den JS-Frameworks was den Einstiegsaufwand verringert.

## Performance und Hosting

#### Asp.Net Web App (MVC/Razor Pages):

Klassische ASP.NET Web Apps sind sehr performant, da bei jeder Benutzerinteraktion nur die jeweils benötigten Seiten oder Inhalte serverseitig generiert und an den Client gesendet werden. Dadurch entstehen sehr kurze Ladezeiten ohne hohe Clientlast. Das Hosting ist unkompliziert, da die Anwendung auf nahezu jedem Webserver betrieben werden kann.

#### Blazor Sever:

Nutzt SignalR für die Kommunikation zwischen Client und Server. Dadurch werden UI-Änderungen in Echtzeit synchronisiert. Dies erfordert zwar eine konstante Serververbindung, bietet aber dafür eine ständig aktuelle Version der Web-App an. Die Performance ist im Normalfall hoch, braucht aber mehr Serverleistung, weil die Clients eine ständige Verbindung zum Sever haben. Das Hosting ist wie bei der klassischen Variante sehr unkompliziert da Blazor auf jedem Server mit einer .NET-Umgebung bereitgestellt werden kann.

#### Blazor WebAssembly:

Blazor WebAssembly wird vollständig im Browser des Benutzers ausgeführt, wodurch die Anwendung nach dem erstmaligen Laden weiter unabhängig von Server arbeitet. Dies entlastet den Server und ermöglicht auch Offlinezugriff. Allerdings kann das zu einer höheren Ladezeit führen, weil die ganze Seite im Browser heruntergeladen werden muss. Das Hosting ist wieder sehr unkompliziert, weil nur statische Dateien wie HTML, CSS und JS-Files bereitgestellt werden.

## Entscheidung für Blazor Server

Nach Absprechung und Recherche zu den Möglichkeiten haben wir uns für Blazor Sever entschieden. Diese Variante vereint die Vorteile einer modernen Webentwicklung mit der einfachen Handhabung einer serverseitigen Architektur.

Unsere Gründe im Detail:

* Einheitliche Programmiersprache: Frontend/Backend vollständig in C#
* Geringerer Einarbeitungsaufwand: Bestehendes C# Vorwissen
* Hohe Interaktivität: Echtzeit Daten werden angezeigt (Verbindung zum Server)
* Einfaches Hosting: Hosting auf jedem Gerät möglich
* Gute Dokumentation: Aktive Community und gute Tutorials im Internet

#### MudBlazor

Für die Gestaltung der Benutzeroberfläche haben wird uns für MudBlazor entschieden. MudBlazor ist ein UI-Famework für Blazor, das auf dem Material-Design-Standard basiert und für moderne Webanwendungen entwickelt wurde.

* Umfangreiche Komponentenbibliothek
* Konsistentes Design
* Einfache Integration mit Blazor
* Aktive Community und Weiterentwicklung
* Erweiterbar