Kriterienkatalog für die Architekturauswahl

Mit diesem Kriterienkatalog sollen Software-Architekten anhand der vorliegenden Problemstellung einen webbasierten Architekturvorschlag (MPA, SPA, Fullstack-Architektur) ableiten können. Entstanden ist das Ganze im Rahmen der Bachelorarbeit „Ist die Trennung zwischen Frontend und Backend in webbasierten Systemen noch zeitgemäß?“ von Michael Mertl. Die Kriterien sind dabei aus einer umfangreichen und internationalen Literaturrecherche zu den verschiedenen Ansätzen entstanden, indem die Vor- und Nachteile dieser ermittelt wurden und dementsprechend als Grundlage dienten für die vergebenen Punkte. Außerdem wurde eine praxisorientierte Qualitätsprüfung in Form von Experteninterview bei der Firma XITASO durchgeführt, um den Kriterienkatalog zu verbessern. Ebenso wurden Experimente in den drei Architekturen durchgeführt, um einen direkten Vergleich vorzunehmen und den Kriterienkatalog zu verbessern.

Funktionsweise:

Bitte nur die Kriterien ankreuzen und gewichten, die die vorliegende Problemstellung erfordern. Am Ende die Gesamtpunktzahl ausrechnen und die Architektur mit der höchsten Punktzahl ist diejenige, die am besten geeignet ist, für die vorliegende Problemstellung. Jede Architektur hat für jedes Kriterium vorweg eine Punktzahl bekommen (1 = nicht/schlecht erfüllt, 3 = teilweise erfüllt, 6 = vollständig erfüllt), die aus der Literaturrecherche anhand der Vor- und Nachteile entstanden sind und durch Experteninterviews und Experimente evaluiert wurden. Diese verrechnet mit dem Gewicht ergibt die Punktzahl für das jeweilige Kriterium. Die Gewichte, die für die einzelnen Kriterien vergeben werden, müssen am Ende 100% entsprechen, um eine mathematisch korrekte Berechnung zu gewährleisten.

Kriterium	Multi-Page-Architektur	Single-Page-Architektur	Fullstack-Architektur (orientiert an Blazor Server)	Gewicht	Gewollt?
Autorisierung mit OAuth 2.0 -> ist die Umsetzung der Autorisierung mit OAuth 2.0 in der Architektur einfach?	Die Autorisierung muss nur in einer Anwendung umgesetzt werden, da bei der MPA nur eine gebaut wird. -> Punktzahl: 6	Die Autorisierung muss in beiden Anwendungen (dem getrennten Frontend und Backend) umgesetzt werden. So entsteht ein Mehraufwand. -> Punktzahl: 3	Die Autorisierung muss nur in einer Anwendung umgesetzt werden, da bei Blazor Server nur eine gebaut wird. -> Punktzahl: 6	Gewicht: MPA: SPA: Fullstack:	☐
Benutzererfahrung -> bietet die Architektur eine angenehme Benutzererfahrung beim Laden neuer Inhalte?	Jede Benutzerinteraktion führt zum neuen Laden der gesamten Seite, welche die neuen Inhalte dann bereitstellt. Zusätzlich werden die Inhalte im Vergleich zur Fullstack-Architektur langsamer geladen, jedoch genauso schnell wie bei der SPA. -> Punktzahl: 1	Benutzerinteraktionen lösen keine ganzen Seitenaufrufe mehr aus und es können Animationen genutzt werden, beim Warten auf neue Inhalte. Dennoch werden diese Inhalte im Vergleich zur Fullstack-Architektur langsamer geladen. -> Punktzahl: 3	Benutzerinteraktionen lösen keine ganzen Seitenaufrufe mehr aus und es können Animationen genutzt werden, beim Warten auf neue Inhalte. -> Punktzahl: 6	Gewicht: MPA: SPA: Fullstack:	☐

Benutzererfahrung -> bietet die Architektur eine angenehme Benutzererfahrung beim Betätigen von Browsertasten (z.B. Vorwärts/Rückwärts Schaltfläche)?	Bei der MPA liegen viele HTML-Seiten vor, daher stellt der Browser bereits von selbst sicher, dass die verschiedenen Browsertasten richtig funktionieren. -> Punktzahl: 6	Bei der SPA liegt nur eine HTML-Seite vor auf der mit JS navigiert wird. Das Standardverhalten des Browsers kann nicht direkt genutzt werden und muss durch Programmierung nutzbar gemacht werden. -> Punktzahl: 1	Bei Blazor Server liegt dasselbe Problem wie bei der SPA vor. Die .NET-Umgebung bringt eine Klasse (namens „NavigationManager“) mit, die es sehr einfach macht, das Standardverhalten des Browsers zu nutzen und richtig zu programmieren. -> Punktzahl: 3	Gewicht: MPA: SPA: Fullstack:	☐
Browseranforderung -> muss der Browser JavaScript fähig sein für die Webanwendung?	Nein, da die MPA auch komplett ohne JS funktionieren würde. -> Punktzahl: 6	Ja -> Punktzahl: 1	Ja -> Punktzahl: 1	Gewicht: MPA: SPA: Fullstack:	☐
Entwicklererfahrung -> welches Fachwissen wird für die Architektur benötigt?	Es muss nur eine Anwendung entwickelt werden. In dieser wird aber Fachwissen im Frontend (HTML, CSS und JS) und Backend (z.B. C#) benötigt.	Es werden zwei Anwendungen entwickelt. Für die Kommunikation dazwischen muss zusätzlich eine API entwickelt werden. Es wird also Fachwissen im	Es muss nur eine Anwendung entwickelt werden. In dieser wird teilweise Wissen im Frontend und Backend benötigt. Hier wird kein JavaScript benötigt, lediglich Wissen in C#, HTML und CSS.	Gewicht: MPA: SPA: Fullstack:	☐

	-> Punktzahl: 3	Frontend (HTML, CSS und JS) und Backend (z.B. C#) benötigt. -> Punktzahl: 3	-> Punktzahl: 6		
Entwicklungszeit -> wie lange dauert die Entwicklung einer Webanwendung mit dieser Architektur?	In der Regel schneller als bei der SPA, da nur eine Anwendung entwickelt werden muss und auf die nötige API für die Kommunikation wie bei einer SPA verzichtet werden kann. Durch die Codetrennung von Frontend und Backend, aber in der Regel langsamer als die Fullstack-Architektur. -> Punktzahl: 3	In der Regel langsamer als bei der MPA und der Fullstack-Architektur, da zwei Anwendungen geschrieben werden und eine API für die Kommunikation entwickelt werden muss. -> Punktzahl: 1	In der Regel schneller als bei der SPA, da nur eine Anwendung entwickelt werden muss und auf die nötige API für die Kommunikation wie bei einer SPA verzichtet werden kann. -> Punktzahl: 6	Gewicht: MPA: SPA: Fullstack:	☐
Erweiterbarkeit -> ist die Architektur in der Zukunft leicht erweiterbar	Leicht erweiterbar, Frontend und Backend sind aber etwas stärker	Leicht erweiterbar, Frontend und Backend können isoliert erweitert	Es wird eine Anwendung entwickelt, in der keine saubere Codetrennung	Gewicht: MPA: SPA:	☐

(in Bezug auf neue Features)?	verknüpft, da diese in einer Anwendung sind. -> Punktzahl: 3	werden, da zwei Anwendungen vorliegen. -> Punktzahl: 6	zwischen Frontend und Backend erzwungen wird, wie bei der MPA oder SPA, dies kann die Erweiterbarkeit erschweren. -> Punktzahl: 3	Fullstack:	
Offlinefähigkeit -> ist die Architektur in der Lage, die Webanwendung auch offline nutzbar zu machen?	Nein -> Punktzahl: 1	Ja (solange keine neue Daten-Anfrage angestoßen wird) -> Punktzahl: 6	Nein -> Punktzahl: 1	Gewicht: MPA: SPA: Fullstack:	☐
Performanz -> wie schnell ist das initiale Laden der Webanwendung?	Lädt initial schnell, da nur eine HTML-Seite (diese kann statisch oder dynamisch sein) geladen werden muss. Dennoch dauert das initiale Laden im Vergleich zur Fullstack-Architektur länger. -> Punktzahl: 3	Erster Ladezyklus dauert sehr lang, da die gesamte Anwendung geladen wird. -> Punktzahl: 1	Lädt initial schnell, da nur eine statische HTML-Seite und zusätzlich eine JavaScript-Bibliothek von Microsoft (diese ist verantwortlich für die Übertragung der Benutzer-interaktionen an den Server) übertragen werden muss. -> Punktzahl: 6	Gewicht: MPA: SPA: Fullstack:	☐

Performanz -> wie schnell ist das Nachladen neuer Inhalte (ohne die Verwendung von zusätzlichen Caching-Mechanismen)?	Eine Anfrage nach neuen Daten fordert eine neue HTML-Seite (wieder statisch oder dynamisch) vom Server an. Der Server stellt bzw. rendert die neue HTML-Seite und gibt diese an den Client zum Anzeigen zurück. Lädt genauso schnell wie initial und wie die SPA, aber langsamer als die Fullstack-Architektur. -> Punktzahl: 3	Eine Anfrage nach neuen Daten stößt eine Anfrage (z.B. HTTP-Anfrage) an den Server an. Dieser sendet dann eine Antwort (z.B. HTTP-Antwort) mit den benötigten Daten zurück. Die neuen Daten werden dann am Client verarbeitet und es wird nur der Seitenteil neu gerendert, der von den neuen Daten betroffen ist. Lädt genauso schnell wie die MPA, aber langsamer als die Fullstack-Architektur. -> Punktzahl: 3	Eine Anfrage nach neuen Daten wird von der JavaScript-Bibliothek von Microsoft über die SignalR dem Server gemeldet. Dieser beschafft sich die neuen Daten und tauscht den betroffenen Teil der HTML-Seite auf dem virtuellen Abbild des Servers aus. Dieser geänderte Teil der Seite wird über die SignalR-Verbindung an den Client geschickt, wo er ebenfalls ausgetauscht wird und somit der neue Inhalt einem Anwender angezeigt wird. -> Punktzahl: 6	Gewicht: MPA: SPA: Fullstack:	☐
Sicherheit -> wie anfällig ist die Architektur für die ungewollte Offenlegung	Die MPA ist wie Blazor Server in der Lage, die vertraulichen Daten auf dem Server zu	Die SPA speichert viele Daten am Client und ist daher anfällig für bösartiges JavaScript,	Blazor Server speichert im Gegensatz zu der SPA die vertraulichen Daten auf dem Server. Durch die Vermischung	Gewicht: MPA: SPA: Fullstack:	☐

von vertraulichen Daten durch einen Entwickler?	speichern. Ebenfalls liegt hier beim Entwickeln eine strikte Trennung zwischen Frontend und Backend vor. Daher ist es beim Entwickeln leichter, keine vertraulichen Daten ungewollt am Client zu verarbeiten oder offenzulegen. -> Punktzahl: 6	welches durch Eingaben eingeschleust werden kann. Der Entwickler muss also beim Entwickeln aufpassen, dass er keine vertraulichen Daten am Client preisgibt. -> Punktzahl: 1	von Frontend und Backend, kann es jedoch vorkommen, dass beim Entwicklungs-prozess unbeabsichtigt vertrauliche Daten auf dem Client verarbeitet werden, da die Anwendung unübersichtlich werden kann. -> Punktzahl: 3		
Serverbelastung -> wie stark wird der Server belastet (ohne die Verwendung von zusätzlichen Caching-Mechanismen)?	Hoch, da jede Anfrage nach neuen Daten eine neue HTML-Seite anfordert. Diese muss jedes Mal neu vom Server bereitgestellt bzw. generiert werden. Der Zustand und die Logik der MPA können am Client und am Server	Niedrig, da jede Anfrage nach neuen Daten nur die genau benötigte Datenmenge vom Sever anfordert und dieser somit nicht so stark belastet wird als bei der MPA. Der Zustand und die Logik in der SPA werden am Client verarbeitet und belastet	Mittel, da jede Anfrage nach neuen Daten wie bei der SPA nur die genau benötigte Datenmenge vom Server anfordert. Die Grundbelastung des Servers ist jedoch höher als bei der MPA oder der SPA, da bei Blazor Server die Logik und die Zustände am Server	Gewicht: MPA: SPA: Fullstack:	☐

	(oder auch in Kombination) verarbeitet werden. Dadurch kann der Server dann zusätzlich belastet werden. -> Punktzahl: 1	somit den Server nicht. Dennoch stellt over- und under-fetching eine Herausforderung dar und muss vermieden werden, sonst wird der Server unnötig belastet. -> Punktzahl: 3	verwaltet werden müssen und diesen somit belasten. -> Punktzahl: 3		
Skalierbarkeit -> ist die Architektur in der Lage horizontal oder vertikal zu skalieren, um mit steigender Anzahl von Benutzern und Anfragen umzugehen?	Bei der MPA kann der Client oder auch der Server die Zustands-verwaltung übernehmen. Hier besteht also die Möglichkeit eine horizontale oder auch eine vertikale Skalierung problemlos durchzuführen. -> Punktzahl: 6	Bei der SPA übernimmt der Client die Zustands-verwaltung, somit kann der Server zustandslos betrieben werden. Hier besteht also die Möglichkeit eine horizontale oder auch eine vertikale Skalierung problemlos durchzuführen. -> Punktzahl: 6	Bei Blazor Server übernimmt der Server die Zustands-verwaltung. Eine vertikale Skalierung ist problemlos möglich. Eine horizontale Skalierung ist jedoch aufgrund dessen und der nötigen Verwaltung der SignalR-Verbindungen über mehrere Serverinstanzen hinweg, eine größere Herausforderung als bei der MPA oder SPA. -> Punktzahl: 3	Gewicht: MPA: SPA: Fullstack:	☐

Suchmaschinen-optimierung (SEO) -> ist die Architektur in der Lage, die Webanwendung für SEO nutzbar zu machen?	Es liegen viele HTML-Seiten vor, deshalb funktioniert die SEO sehr gut. -> Punktzahl: 6	Es liegt nur eine HTML-Seite vor, deshalb funktioniert SEO nicht ohne weitere Technologien. -> Punktzahl: 1	Es liegt nur eine HTML-Seite vor, deshalb funktioniert SEO nicht ohne weitere Technologien. -> Punktzahl: 1	Gewicht: MPA: SPA: Fullstack:	☐
Gesamtergebnis				100%	