Curriculum für

Certified Professional for Software Architecture (CPSA)®
Advanced Level

Modul
WEB

Web-Architekturen

Version 2015.2-DE; 20. April 2020

Inhaltsverzeichnis

This version of this document has been produced with comments (like this one) enabled. It is **NOT** intended for public distribution or publication, but primarily for internal iSAQB purposes.

Einführung: Allgemeines zum iSAQB Advanced Level

Was vermittelt ein Advanced Level Modul?

- Der iSAQB Advanced Level bietet eine modulare Ausbildung in drei Kompetenzbereichen mit flexibel gestaltbaren Ausbildungswegen. Er berücksichtigt individuelle Neigungen und Schwerpunkte.
- Die Zertifizierung erfolgt als Hausarbeit. Die Bewertung und mündliche Prüfung wird durch vom iSAQB benannte Experten vorgenommen.

Was können Absolventen des Advanced Level (CPSA-A)?

CPSA-A-Absolventen können:

- eigenständig und methodisch fundiert mittlere bis große IT-Systeme entwerfen
- in IT-Systemen mittlerer bis hoher Kritikalität technische und inhaltliche Verantwortung übernehmen
- Maßnahmen zur Erreichung von Qualitätsanforderungen konzeptionieren, entwerfen und dokumentieren sowie Entwicklungsteams bei der Umsetzung dieser Maßnahmen begleiten
- architekturrelevante Kommunikation in mittleren bis großen Entwicklungsteams steuern und durchführen

Voraussetzungen zur CPSA-A-Zertifizierung

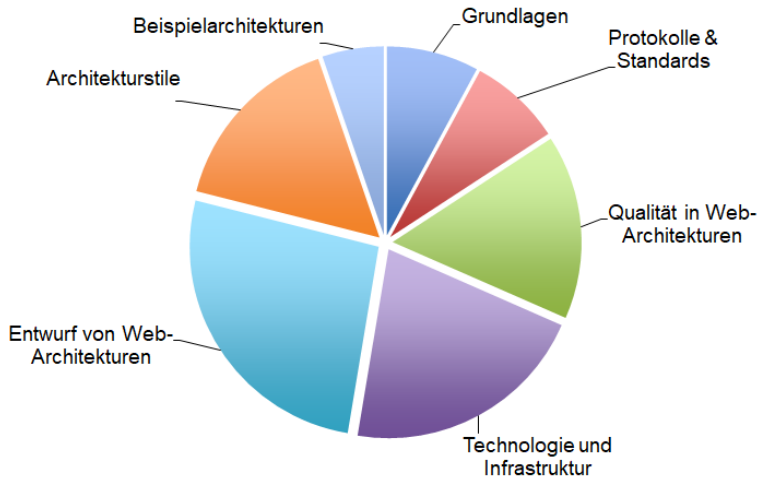
- erfolgreiche Ausbildung und Zertifizierung zum Certified Professional for Software Architecture, Foundation Level® (CPSA-F)
- mindestens drei Jahre Vollzeit-Berufserfahrung in der IT-Branche; dabei Mitarbeit an Entwurf und Entwicklung von mindestens zwei unterschiedlichen IT-Systemen
  - Ausnahmen sind auf Antrag zulässig (etwa: Mitarbeit in Open-Source-Projekten)
- Aus- und Weiterbildung im Rahmen von iSAQB-Advanced-Level-Schulungen im Umfang von mindestens 70 Credit Points aus mindestens drei unterschiedlichen Kompetenzbereichen
  - bestehende Zertifizierungen (etwa Sun/Oracle Java-Architect, Microsoft CSA) können auf Antrag angerechnet werden
- erfolgreiche Bearbeitung der CPSA-A-Zertifizierungsprüfung

Grundlegendes

Struktur des Lehrplans und empfohlene zeitliche Aufteilung

Inhalt	Empfohlene Mindestdauer (min)
1. Grundlagen	90
2. Protokolle und Standards	90
3. Architekturstile	180
4. Technologie und Infrastruktur	240
5. Entwurf von Web-Architekturen	300
6. Qualität in Web-Architekturen	180
7. Beispielarchitekturen	60
Summe	1140 (19h)

Zeitliche Aufteilung der Themengebiete



Bitte sowohl die oben angegebene Tabelle als auch das beiliegende Excel-Dokument entsprechend anpassen und das Pie-Chart als "zeitaufteilung.png" nach `../images/01-basics` exportieren

= = =

Please adjust the table above as well as the excel document according to your curriculum and export the pie chart as "chronological_breakdown.png" to `../images/01-basics`.

Bitte die **?** durch die Anzahl der Tage sowie die erreichbaren Punkte ersetzen.

Dauer, Didaktik und weitere Details

Die unten genannten Zeiten sind Empfehlungen. Die Dauer einer Schulung zum Modul WEB sollte mindestens 3 Tage betragen, kann aber länger sein. Anbieter können sich durch Dauer, Didaktik, Art und

Aufbau der Übungen sowie der detaillierten Kursgliederung voneinander unterscheiden. Insbesondere die Art der Beispiele und Übungen lässt der Lehrplan komplett offen.

Lizenzierte Schulungen zu WEB tragen zur Zulassung zur abschließenden Advanced-Level-Zertifizierungsprüfung folgende Credit Points) bei:

Methodische Kompetenz:	0 Punkte
Technische Kompetenz:	30 Punkte
Kommunikative Kompetenz:	0 Punkte

Voraussetzungen

Teilnehmerinnen und Teilnehmer **sollten** folgende Kenntnisse und/oder Erfahrung mitbringen:

- CPSA-F und alle damit verbundenen Voraussetzungen
- Erfahrungen mit verteilten Systemen – idealerweise Web-Anwendungen
- Grundkenntnisse in Web-Technologien HTML, CSS, JavaScript sowie mindestens ein serverseitiges Framework

Kenntnisgruppen sowie Voraussetzungen bitte entsprechend ausformulieren!

Gliederung des Lehrplans

Die einzelnen Abschnitte des Lehrplans sind gemäß folgender Gliederung beschrieben:

- **Begriffe/Konzepte**: Wesentliche Kernbegriffe dieses Themas.
- **Unterrichts-/Übungszeit**: Legt die Unterrichts- und Übungszeit fest, die für dieses Thema bzw. dessen Übung in einer akkreditierten Schulung mindestens aufgewendet werden muss.
- **Lernziele**: Beschreibt die zu vermittelnden Inhalte inklusive ihrer Kernbegriffe und -konzepte.

Dieser Abschnitt skizziert damit auch die zu erwerbenden Kenntnisse in entsprechenden Schulungen.

Ergänzende Informationen, Begriffe, Übersetzungen

Soweit für das Verständnis des Lehrplans erforderlich, haben wir Fachbegriffe ins iSAQB-Glossar aufgenommen, definiert und bei Bedarf durch die Übersetzungen der Originalliteratur ergänzt.

1. Grundlagen

Dauer: 90 Min.	Übungszeit: keine

1.1. Begriffe und Konzepte

Web-Browser, Web-Server, Client, Server, Proxy, Request, Response, Barrierefreiheit, Basic-Auth, Intranet vs. Internet, Redirect, TLS, Prepocessor, Hypermedia, Statelessness, Pattern Libraries, Frontend Components

1.2. Lernziele

In diesem Abschnitt sollen technologieunabhängig Konzepte aus dem Web-Umfeld behandelt werden.

Was sollen die Teilnehmer können?

- Die Teilnehmer können den typischen Request-/Response-Verlauf erläutern, der bei der Eingabe einer Adresse in der Adresszeile des Browsers bzw. beim Absenden eines Formulars abläuft.
- Teilnehmer können den Request-/Response-Verlauf auf typische Komponenten im Web-Umfeld abbilden: Client, Server, Proxy, Reverse Proxy, Load-Balancer, DNS-Server, Framework, eigene Applikationslogik (Servlet, PHP-Script, Rails-Controller).
- Die Teilnehmer können den Unterschied zwischen GET und POST-Request erläutern.
- Die Teilnehmer können schematisch die typische Client-Server-Infrastruktur von Web-Architekturen skizzieren.
- Teilnehmer können den allgemeinen Aufbau von HTTP-Requests erläutern: Header (mit Hostname, Content-Type usw.), Content.
- Teilnehmer können den allgemeinen Aufbau von HTTP-Responses erläutern: Response mit Statuszeile, Header und Content.
- Die Teilnehmer können den Aufbau einer URI erläutern und die Komponenten in der Request-Verarbeitung, die tendenziell dafür zuständig sind, sie zu interpretieren.

Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen, dass sich hinter dem Oberbegriff Web-Applikation sowohl technisch als auch fachlich grundlegend verschiedene Arten von IT-Systemen verbergen können, die verschiedene Architekturen erfordern
- Die Teilnehmer verstehen, dass Web-Applikationen nicht auf die Verwendung durch Browser beschränkt sind, sondern können auch durch andere Clients genutzt werden
- Die Teilnehmer verstehen den Unterschied zwischen inhaltlicher bzw. struktureller und darstellungsspezifischer Auszeichnung von Inhalten im Sinne der Separation of Concerns.
- Die Teilnehmer verstehen die allgemeinen Anforderungen der Web Content Accessibility Guidelines (WCAG), die unter dem Begriff Barrierefreiheit zusammengefasst sind.
- Die Teilnehmer verstehen den meist für Web-Anwendungen angewandten Flow bei Single Sign On Mechanismen (Redirect, Login, Redirect mit Token, Validierung des Tokens).
- Die Teilnehmer verstehen, dass eine Ressource mehr als eine Datenbank-Entität ist (also z.B. auch eine Prozess-Instanz oder ein Prozessschritt).

- Die Teilnehmer verstehen, dass Hypermedia dazu dient Ressourcen miteinander in Verbindung zu setzten.
- Die Teilnehmer verstehen das Konzept von Präprozessoren zur Übersetzung einer Sprache in eine andere bzw. zur Einbettung von Abhängigkeiten oder zur Erkennung toter Code-Blöcke.

Was sollen die Teilnehmer kennen?

- Die Teilnehmer wissen, dass rechtliche Rahmenbedingungen barrierefreie Oberflächen verlangen können. Z. B. formuliert die „Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz“ etwa konkrete Anforderungen, die von allen Web-Seiten der Bundesverwaltung der Bundesrepublik Deutschland umgesetzt werden müssen.
- Die Teilnehmer kennen einige der Techniken, die zu barrierefreien Inhalten führen, sorgen gleichzeitig dafür, dass diese Inhalte auch leichter maschinell verarbeitet werden können.
- Die Teilnehmer kennen verschiedene Charakteristika von Web-Frameworks:
  - Komponenten- sowie Request-Response-Frameworks nutzen häufig vom MVC-Muster beeinflusste Designs
  - Request-Response-Frameworks sind strukturell meist relativ gleich aufgebaut: Actions nehmen den Request entgegen, führen Applikationslogik aus und rendern zum Schluss die Response meist mit Templating (HTML) oder Objekt-Serialisierung (JSON/XML)
- Die Teilnehmer kennen mit dem Konzept von Pattern-Libraries eine Möglichkeit Frontend-Komponenten zu dokumentieren.

2. Protokolle und Standards

Dauer: 90 Min.	Übungszeit: keine

2.1. Begriffe und Konzepte

URI, URL, URN, HTTP/1.1, HTTP/2, HTTP/3, HTTP-Verben (GET, PUT, POST, DELETE), HTTP-Header, Intermediaries, Caching, Content-Types, Content-Negotiation, TLS, PKI, HTML, DOM, Web-Sockets, Server-Sent Events, CustomElements, Shadow DOM, OAuth 2, OpenID Connect., CORS, Content Security Policy

2.2. Lernziele

Was sollen die Teilnehmer können?

- Die Teilnehmer können Software-Systeme so entwerfen, dass sie an den Schnittstellen die im World Wide Web gebräuchlichen Protokolle effektiv und ressourcenschonend einsetzen.
- Die Teilnehmer können gezielt HTTP-Header verwenden, um der Intrastruktur das Caching zu ermöglichen.
- Die Teilnehmer können benennen, welche HTTP-Header sich auf Caching beziehen und wie sich diese auswirken (Stichwort Validierung vs. Verfallszeitstempel).
- Die Teilnehmer können den Unterschied zwischen den Rollen, die HTML, CSS und JavaScript im Browser spielen erklären.

Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen, dass die Protokolle und die Architektur des Webs technologieunabhängig sind.
- Die Teilnehmer verstehen den Zusammenhang zwischen Server-Adresse und Server-Name und kennen die Auswirkungen für die Erzeugung und Verwendung von URLs, die sowohl System-intern als auch -extern verwendet werden sollten.
- Die Teilnehmer verstehen den Zusammenhang zwischen dem Transport-Protokoll (TCP, UDP) und dem Applikationsprotokoll (HTTP/1.1, HTTP/2 und HTTP/3 & QUIC).
- Die Teilnehmer verstehen, wie eine Namensauflösung per DNS-Lookup funktioniert.
- Die Teilnehmer verstehen, welche Auswirkungen die Cache-Control-Header auf Intermediaries haben.
- Die Teilnehmer verstehen die wesentlichen Eigenschaften des HTTP-Protokolls und können diese erklären.
    - Das HTTP-Protokoll ist ein zustandsloses Request-Response Applikationsprotokoll.
    - HTTP /1.1 ist ein textbasiertes serielles Applikationsprotokoll
    - HTTP/2 multiplexed viele Requests/Responses gleichzeitig über eine TCP-Verbindung.
    - HTTP/3 tauscht TCP gegen UDP und ergänzt die fehlende zuverlässige Verbindung in Form vom QUIC auf der Applikationsprotokollebene.
    - Die Teilnehmer wissen, dass HTTP/3, HTTP/2 und HTTP/1.1 weitestgehend dieselbe Semantik auf Grundlage unterschiedlicher Leitungsprotokolle implementie-ren.
    - Das HTTP-Protokoll sieht dazwischengeschaltete Verarbeitungsprozesse (Inter-mediaries) ausdrücklich vor.

  - Der Client identifiziert gegenüber dem Server eine Ressource über einen URI mit dem Schema http oder https. HTTP-Verben legen dabei die Art des Zugriffs fest, die Verben besitzen Semantik.
  - Der Server antwortet mit standardisierten Status-Codes.
  - HTTP-Header werden für weitere Dienste, Metadaten oder Erweiterungen genutzt.
  - Diverse HTTP-Header legen gemeinsam fest, ob Responses gecachet werden dür-fen.
  - Verschiedene Repräsentationen der gleichen Ressource werden über Medientypen (Content-Types) identifiziert und können zwischen Client und Server über Con-tent-Negotiation ausgehandelt werden.
  - Kompression der Response-Daten kann (und sollte) ebenfalls über einen analogen Mechanismus ausgehandelt werden.
  - Das HTTP-Protokoll bietet Raum für unterschiedliche Verfahren zur Authentifizie-rung des Clients gegenüber dem Server (z.B. Basic-Authentifizierung).
  - Cookies erweitern das Protokoll um einen Mechanismus zur Annotierung der Browser-Sitzung mit beliebigen Informationen.
  - Auf Cookies aufbauend kann das an sich zustandslose Protokoll mittels Serverseiti-gen Sessions statusbehaftet verwendet werden.
  - Teilnehmer verstehen die interne Struktur von HTTP-Requests und -Responses.
  - Die Teilnehmer verstehen, dass das Protokoll TLS auf der Transport-Ebene ver-schlüsselt, bei der eine symmetrische Verschlüsselung erfolgt.
  - Der Schlüssel wird asymmetrisch z.B. mit Zertifikaten für den Server und den Cli-ent ausgehandelt.
  - Zertifikate des Clients können zur Authentifizierung genutzt werden.
  - Zertifikate werden mittels Public-Key Infrastrukturen (PKI) gemanaget.
- Die Teilnehmer verstehen, dass es für unterschiedliche Anforderungen unter-schiedliche Authentifizierungsmechanismen gibt.
- Die Teilnehmer verstehen, dass man Struktur der Informationen von deren Darstellung trennen sollte.
- Die Teilnehmer verstehen, wie ein Redirect verarbeitet wird.
- Die Teilnehmer verstehen den Begriff Idempotenz.
- Die Teilnehmer verstehen die Eigenschaft Safety. Sie wissen, dass ein HTTP-GET safe ist und daher die Response des Servers auf ein GET prinzipiell gecachet werden darf.
- Die Teilnehmer verstehen, warum ein Browser beim „Erneut Laden“ bei HTTP-POST eine separate Bestätigung verlangt.
- Die Teilnehmer verstehen, wie Web-Anwendungen realisiert sein müssen, um solche Rück-fragen des Browsers weitgeh end zu vermeiden.
- Die Teilnehmer verstehen, was HTTPS bedeutet und wie es funktioniert.
- Die Teilnehmer verstehen, welche Auswirkung es hat, eine TLS-Verbindung zu terminieren.
- Die Teilnehmer wissen, dass bei HTTP Basic-Authentication Benutzername und Passwort im Klartext übertragen werden.
- Die Teilnehmer wissen, dass vertrauliche Informationen im Web-Umfeld am besten auf dem Transportweg verschlüsselt übertragen werden sollten. Die Verschlüsselung auf dem Transportweg erfolgt über Transport Level Security (TLS).

- Die Teilnehmer verstehen die Auswirkungen von TLS für das Caching.

Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die gängigen HTTP-Status-Codes und wissen, welche Ursache un-terstellt und welche Reaktion darauf üblicherweise erwartet wird.
- Die Teilnehmer verstehen die Verantwortung und Aufgaben der Protokolle und Komponen-ten im Web-Umfeld:
  - URIs identifizieren Ressourcen, URLs lokalisieren sie zusätzlich bei einer Authori-ty,
  - DNS-Server unterstützen bei der Auflösung des Authority-Teils des URI,
  - Das HTTP-Protokoll ist ein generisches Protokoll zum Zugriff auf Ressourcen und stellt Lösungen für einige nicht-fachliche Anforderungen zur Verfügung,
  - Für verschiedene Arten von Daten stehen standardisierte Formate zur Verfügung,
  - HTTP sieht vor, dass Clients und Server das verwendete Format miteinander aus-handeln, wenn mehrere Alternativen existieren.
- Die Teilnehmer kennen das Browser-intern verwendete Document Object Model wissen, dass es durch JavaScript und CSS verändert werden kann.
- Die Teilnehmer kennen verschiedene Datenformate für die Repräsentation von Informati-onen (HTML, XML, JSON, …).
- Die Teilnehmer wissen, wie Cookies zwischen Client und Server ausgetauscht und verwaltet werden.
- Die Teilnehmer kennen das XML Http Request-Object (kurz XHR) sowie die HTML5-fetch API als Grundlage von AJAX-basierten Anwendungen.
- Die Teilnehmer kennen die Möglichkeiten, die sich durch Server-Side-Events ergeben.
- Die Teilnehmer kennen Mechanismen, die die Verwendung der Browser-Buttons Back und Forward erlauben, ohne unerwünschte Nebeneffekte auszulösen.
- Die Teilnehmer kennen die relevanten Standardisierungsgremien wie IETF, IANA, W3C und deren Aufgabenbereiche im Bezug auf Web-Architektur.
- Die Teilnehmer kennen WebSockets.
- Die Teilnehmer kennen OAuth 2 als ein System zur Zugangsdelegation an Dritte.
- Die Teilnehmer kennen OpenID Connect als ein auf OAuth 2 basierendes System für Single Sign on
- Die Teilnehmer kennen CORS als Mechanismus zum Umgang mit Cross-Domain Restriktionen
- Die Teilnehmer kennen die Content Security Policy als Mechanismus, um Cross-Site-Scripting zu erschweren

2.3. Referenzen

[ECMA-262], [Fielding+2000], [HTML-CSS], [Jacobs+2004], [RFC2246], [RFC2616], [RFC3986], [RFC3987], [RFC6265], [Tilkov 2011], [WS-I]

3. Architekturstile

Dauer: 120 Min.	Übungszeit: 60 Min.

3.1. Begriffe und Konzepte

Representational State Transfer (REST), Stateful-BackEnd-Web-Apps, Single-Page-Applikation, Web-Components

3.2. Lernziele

Was sollen die Teilnehmer können?

- Die Teilnehmer können erklären, wie sich der REST-Stil vom Stateful-Backend-Architekturstil unterscheidet.
- Die Teilnehmer können verschiedene Web-Architektur-Stile unterscheiden und bewusst auswählen, welcher für eine Menge von gegebenen Anforderungen und Rahmenbedingun-gen der sinnvollste ist.
- Die Teilnehmer können für ein auch potentiell ihnen unbekanntes Framework bewerten, wie gut es für die Realisierung von Web-Anwendungen gemäß den Anforderungen geeignet ist.
- Die Teilnehmer können verschiedene Architekturstile von Web-Architekturen erläutern und entsprechende Systeme entwerfen:
  - REST-konforme Web-Anwendungen
  - Stateful-Backend-Anwendungen (beispielsweise Komponenten-Orientrierte Ansätze).

Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen die Einschränkungen, die der Architekturstil REST auf den Ent-wurf eines Systems mit sich bringt:
  - Identifizierbare Ressourcen
  - Uniforme Schnittstelle
  - Zustandslose Kommunikation
  - Repräsentationen
  - Hypermedia.
- Die Teilnehmer verstehen die Einschränkungen, die Stateful-Backend-Architekturen auferlegen:
  - Kommunikation läuft immer gegen dieselbe Server-Instanz.
  - Bei Komponenten-Orientierten Ansätzen enthalten die Requests in der Regel ein Kommando oder einen Diskriminator, der entscheidet, welche Komponente für die Verarbeitung zuständig ist. In der Regel sind diese Dispatch-Informationen hart kodiert oder im Hauptspeicher (in der Session) abgelegt.
- Die Teilnehmer verstehen, dass, gleichgültig welcher Architekturstil gewählt ist, der Kern der Fachlichkeit, also beispielsweise sicherheits- und funktionsrelevante Prüfungen oder Berechnungen, stets im Server realisiert werden muss.
- Die Teilnehmer verstehen, wie sich die Reaktionsfreudigkeit von Web-Applikationen über Funktionalität im Client, also über JavaScript im Client (z. B. in Verbindung mit AJAX), verbessern

lässt.

- Die Teilnehmer verstehen die Standard-Features von Single-Page-Frameworks:
  - Databinding (Two-Way und One-Way)
  - Templating
  - Clientseitiges Routing
  - Komponenten-Abstraktion
- Die Teilnehmer verstehen, dass sich mit WebComponents (CustomElements + Shadow-DOM) stark von der Seite isolierte, eigenständige JavaScript-Komponenten entwickeln lassen, die sich per DOM instanziieren lassen.

Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen Mechanismen langlaufende Transaktionen mit entspre-chenden Status-Codes anzunehmen und über geeignete Mechanismen das Ergebnis zu melden, sobald es vorliegt.
- Die Teilnehmer kennen die Auswirkungen von mobilen Clients:
  - stark unterschiedliche Bandbreiten
  - schwankende teilweise sehr hohe Latenz.
- Die Teilnehmer wissen, dass mobile Clients teilweise reduzierte Leistungsfähigkeit im Bezug auf Hauptspeicher und CPU besitzen.
- Die Teilnehmer wissen, dass Ressourcen von mobilen Clients gerade im Hinblick auf limitierte Energiekapazitäten der Endgeräte zu schonen sind.

4. Technologie und Infrastruktur

Dauer: 180 Min.	Übungszeit: 60 Min.

4.1. Begriffe und Konzepte

Client, Server, Proxy, Reverse-Proxy, Content Delivery Networks/CDN, Lastverteilung/Load Balancing, CGI

4.2. Lernziele

Was sollen die Teilnehmer können?

- Die Teilnehmer können serverseitige Web-Anwendungen so entwerfen, dass sie die Infrastruktur effektiv einsetzen.
- Die Teilnehmer können das Laufzeitverhalten eines Systems durch den gezielten Einsatz von Reverse Proxies verbessern.
- Die Teilnehmer können verschiedene Intermediaries benennen und ihre Auswirkungen auf die Architektur erklären.
  - Proxies speichern Responses für den Client, hierfür sieht das HTTP-Protokoll explizite Regeln vor.
  - Reverse Proxies sind Proxies auf der Server-Seite und fungieren als Caches für die Web-Applikation.
  - Load Balancer verteilen Anfragen auf Server-Seite auf unterschiedliche Server.
- Die Teilnehmer kennen verschiedene Skalierungsstrategien (horizontal sowie vertikal) und können die geeignete Skalierungsstrategie auswählen.
  - Vertikale Skalierung durch leistungsfähigere Hardware.
  - Vertikale Skalierung dadurch, dass verschiedene Komponenten – etwa Web- und Applikationsserver – auf getrennten Infrastruktur-Komponenten verteilt werden.
  - Vertikale Skalierung durch Cluster-Lösungen, die simulieren, dass eine Applikation nur auf einem Knoten läuft, obwohl es mehrere Server-Knoten zur Ausführung gibt.
  - Horizontale Skalierung dadurch, dass ein Server-System dupliziert und die Last möglichst zufällig und gleichmäßig auf die vorhandenen Server verteilt wird. Dieses Verfahren muss für horizontale Skalierung potentiell unendlich fortsetzbar sein.

Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen, dass Caching die Last auf den Server-Systemen reduzieren kann.
- Die Teilnehmer verstehen, wie Daten vieler Web-Anwendungen für wenige Benutzer (Client- oder auch Forward-Proxy) oder vieler Benutzer für wenige Web-Anwendungen (Reverse-Proxy) bereitgestellt werden können.
- Die Teilnehmer verstehen, wie Zugriffe auf Ressourcen durch Proxies kontrolliert werden können.
- Die Teilnehmer verstehen, wie mittels Reverse-Proxy Authentifizierungsschemata (beispielsweise von Form-based auf Basic-Auth.) umgeschrieben werden können.
- Die Teilnehmer sollten verstehen, welche Infrastruktur-Komponenten im Request-/Response-Zyklus Einfluss auf den Datendurchsatz und die Latenzzeit haben.

- Die Teilnehmer sollten verstehen, wie Browser-Clients Ressourcen laden und insbesondere HTML-Seiten auswerten und referenzierte Ressourcen anfragen.
- Die Teilnehmer verstehen, wie das Auslagern von Ressourcen in ein Content Delivery Network (CDN) hilft, die eigene Infrastruktur zu entlasten.
- Die Teilnehmer verstehen, dass nicht alle Web-Browser die Standards in gleichem Umfang unterstützen.
- Die Teilnehmer verstehen, dass sich die verschiedenen Web-Server im Bezug auf den Ressourcenverbrauch teilweise stark unterscheiden

Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen verschiedene Standard-Mechanismen, mit denen Web-Server um Applikationslogik erweitert werden können (CGI, Servlets, …).
- Die Teilnehmer kennen verschiedene Ansätze zur Lastverteilung einer Web-Applikation (z.B. DNS-basierte Load-Balancer oder Proxies).
- Die Teilnehmer kennen mit Graceful Degradation und Progressive Enhancement verschiedene Strategien, die Oberfläche an die Fähigkeiten des Web-Browsers anzupassen.
- Die Teilnehmer kennen sowohl für CSS als auch für JavaScript Frameworks, die die Entwicklung erleichtern können.
- Die Teilnehmer wissen, dass unter HTML5 diverse Technologiepakete zusammengefasst werden, die neben neuen Strukturen für HTML auch neue APIs für JavaScript enthalten.
- Die Teilnehmer kennen die Bedeutung von unobtrusive JavaScript: Darunter werden Prinzipien und Praktiken zusammengefasst, die Inhalte und Verhalten deutlich voneinander trennen.
- Die Teilnehmer kennen die Funktionsweise von Web-Firewalls und wissen gegen welche Arten von Angriffen diese schützen.

4.3. Referenzen

[BITV 2011], [Brewer 2004], [Buschmann+1996], [Daswani+2007], [ESI], [HTML5-SSE], [Kopparapu 2002], [OWASP], [Pritchett 2002], [Stoneburner+2004], [Waldo+1994], [WCAG 2008], [Websockets], [W3C-Int]

5. Entwurf von Web-Architekturen

Dauer: 180 Min.	Übungszeit: 120 Min.

5.1. Begriffe und Konzepte

Datenmodellierung, Funktionale Zerlegung, Repräsentationm, Verteiltes System, CAP-Theorem, ACID, Sicherheit von Web-Applikationen, Authentifizierung, Autorisierung, Barrierefreiheit/Accessibility, Internationalisierung/Lokalisierung, HTML, CSS, JavaScript, Trennen von Inhalt, Präsentation und Verhalten, Graceful Degradation, Progressive Enhancement, unobtrusive JavaScript.

5.2. Lernziele

Was sollen die Teilnehmer können?

- Die Teilnehmer können die Belange Repräsentation, Formatierung und Interaktion klar im Markup unterscheiden und beim Entwurf von Web-Anwendungen berücksichtigen (Separation of Concerns).
- Die Teilnehmer können verschiedene Aufgaben der Client-Oberfläche Formaten aus dem Kanon der Web-Standards zuordnen:
  - HTML liefert die Strukturdaten
  - CSS legt die Präsentation fest. Selektoren wählen dabei Elemente aus der HTML-Struktur und weisen ihnen Präsentationseigenschaften zu.
  - JavaScript steuert das Verhalten, verleiht über Event-Handler Interaktivität und ermöglicht asynchrone Serverkommunikation (AJAX).

Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen die Notwendigkeit von Regeln, die beim Einsatz von Request-Response-Frameworks sicherstellen, dass nur erwünschte Funktionalitäten in den Views/Templates implementiert werden.
- Die Teilnehmer verstehen, dass sich die Reaktionsfähigkeit des Systems durch geeigneten Einsatz von asynchroner Verarbeitung oft verbessern lässt.
- Die Teilnehmer verstehen die grundsätzliche Aussage des CAP-Theorems und wissen, dass zwischen Skalierbarkeit/Verfügbarkeit/Verteiltheit und Konsistenz/Aktualität der Daten ein Kompromiss gesucht werden muss.
- Die Teilnehmer verstehen, wie Netzwerk-Fehler in verteilten Systemen Inkonsistenzen oder besondere Fehlerzustände hervorrufen können.
- Die Teilnehmer verstehen, wie sich Architekturen für konsistente Systeme und Systeme mit Partitionstoleranz unterscheiden.
- Die Teilnehmer verstehen, wie Angreifer Sicherheitslücken mit Hilfe von SQL-Injection, per Cross-Site-Scripting (XSS) oder per Client-Side-Request-Forgery (CSRF) ausnutzen können.
- Die Teilnehmer verstehen das Transaktionskonzept ACID – Atomicity, Consistency, Isolation und Durability.
- Die Teilnehmer verstehen das Konzept der Eventual Consistency (konsistent nach einiger Verzögerung).

- Die Teilnehmer verstehen, dass die innere Architektur der Server-Seite zusätzlich zu den fachlichen Aspekten auch durch
  - die Philosophie des verwendeten Frameworks,
  - Strategien zum Caching von statischen und dynamischen Inhalten sowie
  - Strategien zur Skalierung der Applikation getrieben wird.
- Die Teilnehmer verstehen, dass sich für lang laufende Prozesse Integrationsmuster wie Messaging für eine lose Kopplung der Web-Applikation an eine Hintergrundverarbeitung anbieten.
- Die Teilnehmer verstehen, dass jede Anwendung, die aus öffentlichen Netzen erreichbar ist, angegriffen wird.

Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die Muster Command-Query-Separation, CQS, und Command-Query-Responsibility-Separation, CQRS, um sowohl funktionale als auch die nicht-funktionale Anforderungen gezielt zu erfüllen.
- Die Teilnehmer wissen, dass es sich bei der Consistency aus „ACID“ um eine andere Konsistenz handelt als bei der Consistency aus CAP.
- Die Teilnehmer kennen die Grundmechanismen, mit denen im eigenen Technologieportfolio SQL-Injection, Cross-Site-Scripting (XSS) und Client-Side-Request-Forgery (CSRF) verhindert werden.
- Die Teilnehmer kennen für gestellte Qualitätsanforderungen verschiedene Strategien zur Verwaltung von sitzungsbezogenen Daten, z. B. im Cookie, im Hauptspeicher des Servers, in einer Datenbank.
- Die Teilnehmer kennen den Unterschied zwischen Internationalisierung und Lokalisierung.
- Die Teilnehmer kennen verschiedene Strategien mit denen ein Client über das Ergebnis einer Aktivität informiert werden kann, die asynchron auf dem Server gestartet wurde. Z.B.:
  - Polling (Client Pull)
  - HTML5 Server Sent Events
  - Websockets.
- Die Teilnehmer wissen, dass viele serverseitige Frameworks Plug-Ins, analog zum Pipes-And-Filter Muster erlauben, beispielsweise ServletFilter in Java Servlet API basierten Frameworks oder HttpModules in ASP.NET.
- Die Teilnehmer kennen zumindest ein serverseitiges Framework, mit dem sich Web-Anwendungen realisieren lassen.
- Die Teilnehmer kennen unterschiedliche Arten von serverseitigen Web-Frameworks:
  - Komponenten-Frameworks versuchen ein Event-basiertes Programmier-Modell aus dem Bereich der Desktop-Applikationen zu übertragen.
  - Action- oder Request-Response-Frameworks bilden das HTTP-Protokoll unmittelbar ab.
  - Daten-getriebene Frameworks bilden zur Datenerfassung Datenbank-Tabellen in der Oberfläche ab.
- Die Teilnehmer kennen Authentifizierungsmechanismen wie OpenID Connect oder CAS.

5.3. Referenzen

[abbott], [BITV 2011], [Brewer 2004], [Buschmann+1996], [Daswani+2007], [ECMA-262], [Evans 2004], [Fowler 2011], [Hohpe+2003], [HTML5], [HTML-CSS], [HTML5-SSE], [Nottingham 1998], [Olsson 2007], [OWASP], [Pritchett 2002], [Stoneburner+2004], [Waldo+1994], [WCAG 2008], [Websockets], [W3C-Int]

6. Qualität in Web-Architekturen

Dauer: 120 Min.	Übungszeit: 60 Min.

6.1. Begriffe und Konzepte

Sicherheit, Skalierbarkeit, Web-Scale, Verfügbarkeit, Bedienbarkeit, Barrierefreiheit

6.2. Lernziele

Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen die Grundlagen der Risikoanalyse und Bedrohungsmodellierung.
- Die Teilnehmer verstehen die Prinzipien für ein sicheres Software-Design:
  - Principle of Least Privilege: gib Benutzern und Komponenten nur so viele Rechte, wie sie unbedingt benötigen.
  - Defense in Depth: vertraue nicht darauf, dass andere Sicherheitsmaßnahmen funktioniert haben – verwende redundante Prüfungen.
  - Secure by Default: Rechte müssen explizit erteilt werden.
  - Don’t trust anybody: Benutzereingaben und Daten externer Quellen müssen validiert werden.
  - Compartmentalize: Trenne Komponenten, die verschiedene Rechte benötigen voneinander.
  - Fail securely: Falls Fehler auftreten, dürfen keine sicherheitsrelevanten Informationen preisgegeben werden.
- Die Teilnehmer verstehen, dass die Anzahl der Zugriffe im Internet sprunghaft über alle erwarteten Maße hinaus steigen kann.
- Die Teilnehmer verstehen, dass Backend-Systeme (wie z.B. Datenbanken oder Enterprise Informationssysteme) leicht unter Last geraten, wenn die Anzahl der Requests steigt, weil häufig ein Request zu einer Menge von Anfragen ans Backend führt.
- Die Teilnehmer verstehen, dass sich mit verschiedenen Persistenzstrategien (z.B. Relational, Aggregat/Dokumenten-orientiert, Spalten-orientiert, Hierarchisch, Objektorientiert, Graphenorientiert) verschiedene Nutzungsszenarien unterschiedlich gut unterstützen lassen.
- Die Teilnehmer verstehen, dass die Qualität sowohl von der Angemessenheit der ausgewählten Technologie als auch von der konkreten Implementierung abhängen.
- Die Teilnehmer verstehen, dass Internationalisierung mehr bedeutet, als Inhalte in verschiedenen Sprachen anzubieten. Etwa
  - Formate für Zahlen oder Datumsangaben oder die Reihenfolge, in der Worte sortiert werden, hängen vom Kulturkreis ab.
  - Schriftrichtungen können sich unterscheiden.
  - Layouts müssen berücksichtigen, dass Texte in verschiedenen Sprachen unterschiedlich viel Platz beanspruchen.
  - Die rechtlichen Rahmenbedingungen in unterschiedlichen Staaten müssen berücksichtigt werden.
  - Kulturelle Unterschiede wirken sich beispielsweise auf die Wahl von Farben aus.

Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die wesentlichen Kennzahlen, die für den Betrieb einer Web-Anwendung relevant sind:
  - Anzahl von Prozessen / Threads
  - RAM-Verbrauch
  - Durchschnittlicher Ressourcenverbauch pro Request
  - Anzahl offener Connections
  - Durchschnittliche Antwortzeit

7. Beispiele

Dauer: 60 Min.	Übungszeit: keine

7.1. Begriffe und Konzepte

Innerhalb jeder akkreditierten Schulung müssen konrkete Beispiele für unterschiedliche Bereiche von Web-Architekturen vorgestellt, besprochen und bewertet werden:

- Umgang mit Infrastruktur und insbesondere Reverse-Proxies.
- Ein grobes Muster einer Web-Backend-Applikation.
- Ein oder mehrere Beispiele für Single-Page-Framework Features wie Data-Binding oder Client-Side-Routing.
- Beispiele für unterschiedliche Datenbank-Modelle.

Art und Ausprägung der vorgestellten Beispiele können von der Schulung bzw. den Interessen der Teilnehmer abhängen und werden seitens iSAQB nicht vorgegeben.

Referenzen

Dieser Abschnitt enthält Quellenangaben, die ganz oder teilweise im Curriculum referenziert werden.

Aufbau eines Eintrags-Ankers:

```
- [[[label,Text der erscheint]]]
```

ACHTUNG: Die Labels dürfen nur Buchstaben beinhalten, keine Zahlen oder Sonderzeichen. Zudem darf zwischen Label und Text *NUR* das Komma, aber kein Leerzeichen stehen.

= = =

Structure of an anchor:

```
- [[[label,text that will be shown]]]
```

ATTENTION: labels have to be non-numeric. There must be *NO* space between label, komma and text.

A

- [Abbot+2010] Abbott, M. L., M. T. Fisher: The Art of Scalability, Addison-Wesley 2010

B

- [BITV 2011] Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungsgesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0) – http://www.gesetze-im-internet.de/bitv_2_0/BJNR184300011.html
- [Brewer 2004] Brewer, E.: Towards Robust Distributed Systems. Keynote zur PODC (Symposium on Principles of Distributed Computing) 2000. http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf
- [Buschmann+1996] Buschmann, F, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: A System of Patterns. Pattern Oriented Software Architecture. Wiley 1996

C

- [Crockford+2008] Crockford, Douglas: JavaScript – The good Parts. O'Reilly, 2008

D

- [Daswani+2007] Daswani, N., C. Kern, A. Kesavan: Foundations of Security: What Every Programmer Needs to Know. Apress 2007

E

- [ECMA-262] ECMAScript Language Specification – http://www.ecma-international.org/publications/standards/Ecma-262-arch.htm
- [Evans 2004] Evans, E.: Domain Driven Design, Addison Wesley 2004
- [ESI] ESI Language Specification 1.0. W3C Note 2001 - http://www.w3.org/TR/esi-lang

F

- [Fielding+2000] Fielding, R., R. Taylor: Principled Design of the Modern Web Architecture. In Proceedings of the 2000 International Conference on Software Engineering (ICSE 2000), Limerick, Ireland, June 2000 - http://www.ics.uci.edu/~fielding/pubs/webarch_icse2000.pdf
- [Fowler 2011] Fowler, M.: CQRS. http://martinfowler.com/bliki/CQRS.html

H

- [Hohpe+2003] Hohpe, G, Woolf, B: Enterprise Integration Patterns, Addison-Wesley 2003
- [HTML5] Web Hypertext Application Technology Working Group (WHATWG): HTML - Living Standard. http://www.whatwg.org/specs/web-apps/current-work/multipage/
- [HTML-CSS] W3C Standards für das Webdesign – http://www.w3.org/standards/webdesign/htmlcss
- [HTML5-SSE] Server-Sent Events. – http://www.whatwg.org/specs/web-apps/current-work/multipage/comms.html#server-sent-events

J

- [Jacobs+2004] Jacobs, I., N. Walsh: Architecture of the World Wide Web, Volume One. W3C Recommendation 2004, - http://www.w3.org/TR/2004/REC-webarch-20041215/

K

- [Kopparapu 2002] Kopparapu, C.: Load Balancing Servers, Firewalls, and Caches. Wiley 2002

N

- [Nottingham 1998] Nottingham, M: Caching Tutorial. http://www.mnot.net/cache_docs/

O

- [Olsson 2007] Olsson, T.: Graceful Degradation & Progressive Enhancement. http://accessites.org/site/2007/02/graceful-degradation-progressive-enhancement/
- [OWASP] Open Web Application Security Project (OWASP) - https://www.owasp.org/

P

- [Pritchett 2002] Pritchett, D.: BASE an ACID alternative. 2008 - http://queue.acm.org/detail.cfm?id=1394128

R

- [RFC2246] Dirks, T., C. Allen: RFC 2246, The TLS Protocol. http://www.ietf.org/rfc/rfc2246
- [RFC2616] Fielding, R., J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: RFC 2616,

Hypertext Transfer Protocol — HTTP/1.1. https://tools.ietf.org/html/rfc2616

- [RFC3986] Berners-Lee, T., R. Fielding, L. Masinter: RFC 3986, Uniform Resource Identifier (URI): Generic Syntax. http://tools.ietf.org/html/rfc3986
- [RFC3987] Duerst, M., M. Suignard: RFC 3987, Internationalized Resource Identifiers. http://tools.ietf.org/html/rfc3987
- [RFC6265] Barth, A.: RFC 6265, HTTP State Management Mechanism. http://tools.ietf.org/html/rfc6265

S

- [Stoneburner+2004] Stoneburner, G., C. Hayden, A. Feringa: Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A. NIST Special Publication 800-27 Rev A 2004 – http://csrc.nist.gov/publications/nistpubs/800-27A/SP800-27-RevA.pdf

T

- [Tilkov 2011] Tilkov, S.: REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien. Dpunkt 2011

W

- [W3C-Int] W3C Internationalization Activity - http://www.w3.org/International/
- [Waldo+1994] Waldo, J., G. Wyant, A. Wollrath, S. Kendall: A Note on Distributed Computing. Sun Microsystems 1994 – http://labs.oracle.com/techrep/1994/smli_tr-94-29.pdf
- [WCAG 2008] Web Content Accessibility Guidelines Working Group: Web Content Accessibility Guidelines. W3C 2008 – http://www.w3.org/WAI/intro/wcag
- [Websockets] Web sockets - http://www.whatwg.org/specs/web-apps/current-work/multipage/network.html#network
- [WS-I] WS-I Profiles - http://ws-i.org/deliverables/Default.aspx