Curriculum für

CPSA Certified Professional for Software Architecture®

– Advanced Level –

Modul:
WEB

Web-Architekturen

Version 1.3 (Februar 2015)

Inhaltsverzeichnis

0 Einleitung: Allgemeines zum iSAQB-Advanced-Level

0.1 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.

0.2 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 nichtfunktionaler Anforderungen konzeptionieren, entwerfen und dokumentieren. Entwicklungsteams bei der Umsetzung dieser Maßnahmen begleiten.
- Architekturrelevante Kommunikation in mittleren bis großen Entwicklungsteams steuern und durchführen

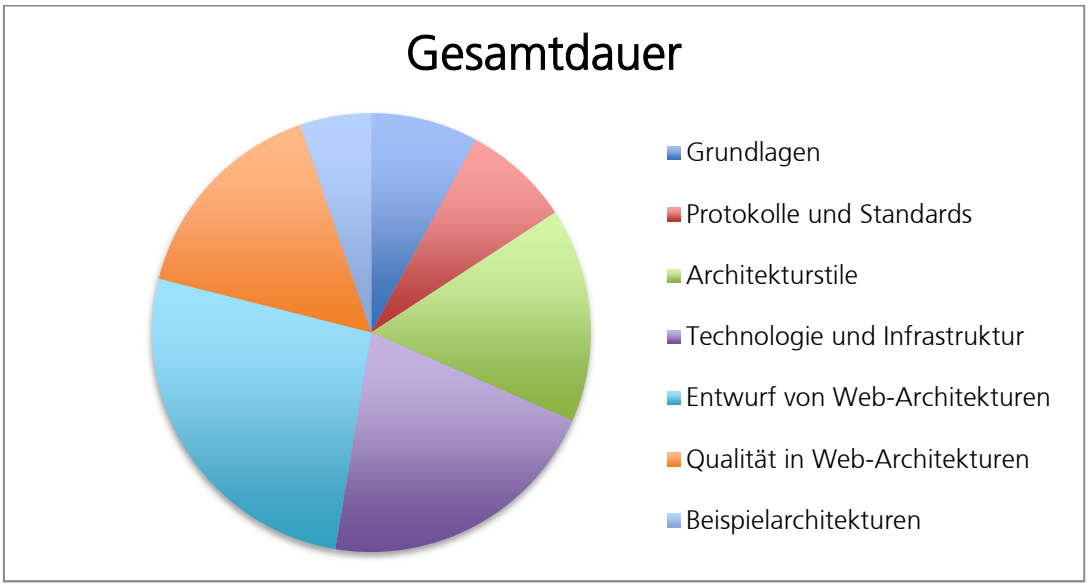
0.3 Voraussetzungen zur CPSA-A-Zertifizierung

- Eine erfolgreiche Ausbildung und Zertifizierung zum CPSA-F (Certified Professional for Software Architecture, Foundation Level)
- Mindestens drei Jahre Vollzeit-Berufserfahrung in der IT-Branche, dabei Mitarbeit an Entwurf und Entwicklung von mindestens zwei unterschiedlichen IT-Systemen
    - Ausnahmen auf Antrag zulässig (etwa: Mitarbeit in OpenSource-Projekten)
- Aus- und Weiterbildung im Rahmen von iSAQB-Advanced-Level-Schulungen im Umfang von mindestens 70 Credit Points aus allen drei unterschiedlichen Kompetenzbereichen (detailliert geregelt in Abschnitt 1.6).
    - Bestehende Zertifizierungen können ggfs. auf Antrag auf diese Credit Points angerechnet werden. Die Liste der aktuellen Zertifikate, für die Credit Points angerechnet werden, ist auf der iSAQB-Homepage zu finden.
    - Sonstige Aus- und Weiterbildungen können auf Antrag beim iSAQB ebenfalls anerkannt werden, sofern ein Bezug zu Software-Architektur vorliegt. Die Entscheidung hierüber trifft im Einzelfall die Arbeitsgruppe „Advanced Level“ des iSAQB.
- Erfolgreiche Bearbeitung der CPSA-A-Zertifizierungsprüfung.

1 Grundlegendes zum Modul Web-Architektur

1.1 Gliederung des Lehrplans für Web-Architektur und empfohlene zeitliche Aufteilung

- Grundlagen (mindestens 90 min)
- Protokolle und Standards (mindestens 90 min)
- Architekturstile (mindesten 180 min)
- Technologie und Infrastruktur (mindestens 240 min)
- Entwurf von Web-Architekturen (mindestens 300 min)
- Qualität in Web-Architekturen (mindestens 180 min)
- Beispielarchitekturen (mindestens 60 min)



1.2 Dauer, Didaktik und weitere Details

Die unten genannten Zeiten sind Empfehlungen. Die Dauer einer Schulung zum Web-Architektur 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.

1.3 Voraussetzungen für das Modul Web-Architektur

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

1.4 Gliederung des Web-Architekturlehrplans

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. Die Lernziele werden differenziert in folgende Kategorien bzw. Unterkapitel:

- Was sollen die Teilnehmer **können**? Diese Inhalte sollen die Teilnehmer nach der Schulung selbständig anwenden können. Innerhalb der Schulung werden diese Inhalte durch Übungen abgedeckt und sind Bestandteil der Modulprüfung Web-Architektur und/oder der Abschlussprüfung des iSAQB-Advanced-Levels.
- Was sollen die Teilnehmer **verstehen**? Diese Inhalte können in der Modulprüfung Web-Architektur geprüft werden.
- Was sollen die Teilnehmer **kennen**? Diese Inhalte (Begriffe, Konzepte, Methoden, Praktiken oder Ähnliches) können das Verständnis unterstützen oder das Thema motivieren. Diese Inhalte sind nicht Bestandteil der Prüfungen, werden in Schulungen thematisiert, aber nicht notwendigerweise ausführlich unterrichtet.
- **Referenzen:** Verweise auf weiterführende Literatur, Standards oder andere Quellen. Eine ausführliche Liste von Büchern und weiteren Quellen findet sich auf der iSAQB-Homepage unter „Fachliche Quellen“.

1.5 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.

1.6 Credit Points für diese Schulungen

Vom iSAQB lizensierte Schulungen gemäß dieses Lehrplans tragen zur Zulassung zur abschliessenden Advanced-Level-Zertifizierungsprüfung folgende Punkte (Credit Points) bei:

Technische Kompetenz: 30 Punkte

2 Einführung in das iSAQB-Zertfizierungsprogramm

Dauer: 15 Min (optional)	Übungszeit: keine

Dieser Abschnitt ist nicht prüfungsrelevant. Falls Teilnehmer bereits CPSA-F zertifiziert sind, kann dieser Abschnitt entfallen.

2.1 Begriffe und Konzepte

iSAQB, Advanced-Level Zertifizierung und Voraussetzung dazu.

2.2 Lernziele

Die Teilnehmer lernen den Kontext des iSAQB-Zertifizierungsprogrammes und der zugehörigen Prüfungen beziehungsweise Prüfungsmodalitäten kennen.

2.2.1 Was sollen die Teilnehmer kennen?

- iSAQB als Verein
- Advanced Level in Abgrenzung zu anderen Level
- Randbedingungen und Vorgehen beim iSAQB Zertifizierungsprogramm

3 Grundlagen

Dauer: 90 Min	Übungszeit: keine

3.1 Begriffe und Konzepte

Web-Browser, Web-Server, Client, Server, Proxy, HTTP, HTML, CSS, JavaScript, URI, Request, Response, Barrierefreiheit, Basic-Auth, Digest-Auth, Intranet vs. Internet, Redirect, TLS, SSL.

3.2 Lernziele

3.2.1 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 Komponente (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.
- Die Teilnehmer kennen die Rollen, die HTML, CSS und JavaScript im Browser spielen.

3.2.2 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-Anwendungen häufig sowohl Online-Transaktions- als auch Online-Analyse-Anwendungsfälle realisieren, für die jeweils unterschiedliche Architektur-Muster zum Tragen kommen.
- 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, wie ein Redirect verarbeitet wird.
- 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 Rückfragen des Browsers weitgehend zu vermeiden.
- Die Teilnehmer verstehen, was HTTPS bedeutet und wie es funktioniert.
- Die Teilnehmer verstehen, welche Auswirkung es hat, eine SSL-Verbindung zu terminieren.
- Die Teilnehmer verstehen den Unterschied zwischen inhaltlicher bzw. struktureller und darstellungsspezifischer Auszeichnung von Inhalten.
- Einige der Techniken, die zu barrierefreien Inhalten führen, sorgen gleichzeitig dafür, dass diese Inhalte auch leichter maschinell verarbeitet werden können.

3.2.3 Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die relevanten Standardisierungsgremien wie IETF, IANA, W3C und deren Aufgabenbereiche im Bezug auf Web-Architektur.
- 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 die allgemeinen Anforderungen, die unter dem Begriff Barrierefreiheit zusammen gefasst sind.
- Komponenten- sowie Action- oder Request-Response-Frameworks nutzen häufig vom MVC-Muster beeinflusste Designs.
- Die Teilnehmer kennen mindestens eine Client-Technologie/eine Markup-Sprache, um Informationen an den Client zu übermitteln und damit die gewünschte Funktionalität zu realisieren.
- Die Teilnehmer wissen, dass HTTP ein textbasiertes Applikationsprotokoll ist.
- Die Teilnehmer kennen die verschiedenen standardisierten Authentisierungsmechanismen: HTTP Basic- bzw. Digest-Authentication.
- 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 (Transport Level Security, TLS) erfolgt über SSL.
- Die Teilnehmer kennen die Auswirkungen von SSL für das Caching.
- Die Teilnehmer kennen WebSockets und SPDY.

4 Protokolle und Standards

Dauer: 90 Min	Übungszeit: keine

4.1 Begriffe und Konzepte

URI, URL, URN und IRI, HTTP, HTTP-Verben (GET, PUT, POST, DELETE), HTTP-Header, Intermediaries, Caching, Idempotenz, Content-Types, Content-Negotiation, SSL/TLS, HTML, DOM, RSS, ATOM.

4.2 Lernziele

4.2.1 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 benennen, welche Art von Anfragen zu welchem Datenverkehr führen.
- Die Teilnehmer können gezielt HTTP-Header verwenden, um der Intrastruktur das Caching zu ermöglichen.
- Die Teilnehmer können für gegebene Anforderungen passende Authentisierungsmechanismen (Basic- und Digest-Authentication) verwenden.
- Die Teilnehmer können Web-Anwendungen entwerfen, in denen die Browser-Buttons Back und Forward generell funktionieren, ohne unerwünschte Nebeneffekte auszulösen.
- Die Teilnehmer können den Kommunikationskanal mit TLS absichern und wissen welcher Authentisierungsmechanismus im Falle einer unverschlüsselten Kommunikation geeignet ist.
- Die Teilnehmer können für gängige Formate abschätzen, wie groß dynamische Inhalte werden können (sowohl im Browser als auch auf dem Transport-Weg).
- Die Teilnehmer können benennen, welche HTTP-Header sich auf Caching beziehen und wie sich diese auswirken (Stichwort Validierung vs. Verfallszeitstempel).
- Die Teilnehmer können Struktur der Informationen von deren Darstellung trennen.
- Die Teilnehmer können die Größenverhältnisse von Nutzlast und Overhead für die Formate einordnen und können anhand zu dem erwartenden Datenaufkommen und der geplanten Verwendung, die passenden Repräsentationsformen auswählen.

4.2.2 Was sollen die Teilnehmer verstehen?

- Die Protokolle und die Architektur des Webs sind technologieunabhängig.
- 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/IP) und dem Applikationsprotokoll (HTTP).
- 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 Text-basiertes und zustandsloses Request-Response Applikationsprotokoll.
    - Das HTTP-Protokoll sieht dazwischengeschaltete Verarbeitungsprozesse (Intermediaries) 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.
- HTTP-Verben, Status-Codes und diverse HTTP-Header legen gemeinsam fest, ob Responses ge-cache-t werden dürfen.
- Verschiedene Repräsentationen der gleichen Ressource werden über Medientypen (Content-Types) identifiziert und können zwischen Client und Server über Content-Negotiation ausgehandelt werden.
- Das HTTP-Protokoll enthält ein Sub-Protokoll zur Authentisierung des Clients gegenüber dem Server (Basic- und Digest-Authentisierung).
- Cookies erweitern das an sich zustandslose Protokoll um einen Mechanismus für statusbehaftete Kommunikation.
- Teilnehmer verstehen die interne Struktur von HTTP-Requests und -Responses.
- Die Teilnehmer verstehen, dass das Protokoll SSL auf der Transport-Ebene verschlüsselt (TLS), bei der eine symmetrische Verschlüsselung erfolgt.
- Der Schlüssel wird asymmetrisch mit Zertifikaten für den Server und den Client ausgehandelt.
- Zertifikate des Clients können zur Authentisierung genutzt werden.

4.2.3 Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die gängigen HTTP-Status-Codes und wissen, welche Ursache unterstellt und welche Reaktion darauf üblicherweise erwartet wird.
- Die Teilnehmer verstehen die Verantwortung und Aufgaben der Protokolle und Komponenten im Web-Umfeld:
  - URIs identifizieren und lokalisieren Ressourcen,
  - 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,
  - Verschlüsselung erfolgt unterhalb der Ebene des HTTP-Protokolls,
  - Für verschiedene Arten von Daten stehen standardisierte Formate zur Verfügung,
  - HTTP sieht vor, dass Clients und Servers das verwendete Format miteinander aushandeln, wenn mehrere Alternativen existieren.
- Die Teilnehmer kennen das Browser-intern verwendete Document Object Model und können mit dessen Hilfe Dokumente im Browser auswerten oder manipulieren.
- Die Teilnehmer kennen verschiedene Datenformate für die Repräsentation von Informationen (HTML, XML, XHTML, JSON, CSV…).
- Die Teilnehmer kennen den Begriff Idempotenz und wissen, für welche HTTP-Verben Requests idempotent sind und für welche die Response des Servers ge-cache-t werden darf.
- Die Teilnehmer wissen, dass Web-Server häufig intern in Komponenten für die Implementierung des HTTP-Servers und für die Implementierung des Anwendungs-Codes unterteilt werden (beispielsweise Web-Container und Applikations-Container).
- Die Teilnehmer wissen, wie Cookies zwischen Client und Server ausgetauscht und verwaltet werden.
- Die Teilnehmer kennen verschiedene Formate zur Realisierung von Feeds wie RSS und Atom.
- Die Teilnehmer kennen das XML Http Request-Object (kurz XHR) als Grundlage von AJAX-basierten Anwendungen.
- Die Teilnehmer kennen die Möglichkeiten, die sich durch Server-Side-Events ergeben.

4.3 Referenzen

[RFC3986]
[RFC3987]
[RFC2616]
[Jacobs+2004]
[Fielding+2000]

[HTML-CSS]
[ECMA-262]
[WS-I]
[Tilkov 2011]
[RFC2246]
[RFC6265]

5 Architekturstile

Dauer: 120 Min	Übungszeit: 60

5.1 Begriffe und Konzepte

Representational State Transfer (REST), Resource-oriented Client Architecture (ROCA), Single-URI-/Stateful-BackEnd-Web-Apps, Single-Page-Applikation, Cross-Compiler-Architekturen, Fat-Client-im-Browser-Apps, Smart-Client-im-Browser-Apps.

5.2 Lernziele

5.2.1 Was sollen die Teilnehmer können?

- Die Teilnehmer können erklären, wie sich der REST-Stil vom Single-URI-/Stateful-BackEnd-Architekturstil unterscheidet.
- Die Teilnehmer können verschiedene Architektur-Stile unterscheiden und bewusst auswählen, welcher für eine Menge von gegebenen Anforderungen und Rahmenbedingungen der sinnvollste ist.
- Die Teilnehmer können die Schnittstellen so entwerfen, dass langlaufende Transaktionen mit entsprechenden Status-Codes angenommen werden und über geeignete Mechanismen das Ergebnis gemeldet wird, sobald es vorliegt.
- Die Teilnehmer können für ein gegebenes Framework bewerten, wie gut es für die Realisierung von Web-Anwendungen gemäß eines vorgegebenen Architektur-Stils geeignet ist.
- Die Teilnehmer können verschiedene Architekturstile von Web-Architekturen erläutern und entsprechende Systeme entwerfen:
    - REST-konforme Web-Anwendungen
    - Single-URI-/Single-Method-Anwendungen
    - Fat-Client-im-Browser-Anwendungen
    - Smart-Client-im-Browser-Anwendungen.

5.2.2 Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen die Einschränkungen, die der Architekturstil REST auf den Entwurf eines Systems mit sich bringt:
    - Identifizierbare Ressourcen
    - Uniforme Schnittstelle
    - Zustandslose Kommunikation
    - Repräsentationen
    - Hypermedia.
- Die Teilnehmer verstehen die Einschränkungen, die Single-URI-/Stateful-Backend-Architekturen auferlegen:
    - Kommunikation läuft über einen zentralen Dispatcher.
    - Requests enthalten 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 abgelegt.
- Die Teilnehmer können beim Einsatz von Werkzeugen zur Generierung von Client-spezifischen Artefakten wie HTML/JavaScript o. ä. aus serverseitigem Quellcode zwischen Vor- und Nachteilen abwägen.
- Die Teilnehmer verstehen, dass, gleichgültig welcher Architekturstil gewählt ist, sicherheitsrelevante Prüfungen stets im Server realisiert werden müssen.
- Die Teilnehmer verstehen, wie sich die Reaktionsfreudigkeit von Web-Applikationen über Funktionalität im Client verbessern lässt. Z. B. über:
    - JavaScript im Client (z. B. in Verbindung mit AJAX)

- Proprietäre Erweiterungen wie Adobe Flash oder Microsoft Silverlight
- Smart-Client Applikationen.

5.2.3 Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die Auswirkungen von mobilen Clients:
  - stark unterschiedliche Bandbreiten
  - teilweise sehr geringe Bandbreiten
  - schwankende Bandbreiten.
- Die Teilnehmer wissen, dass mobile Clients teilweise reduzierte Leistungsfähigkeit im Bezug auf Hauptspeicher und CPU besitzen.

6 Technologien und Infrastruktur

Dauer: 180 Min	Übungszeit: 60 min

6.1 Begriffe und Konzepte

Client, Server, Proxy, Reverse-Proxy, Content Delivery Networks/CDN, Lastverteilung/Load Balancing, CGI, FastCGI, Servlets, ActiveServerPages, PHP-Seiten…

6.2 Lernziele

6.2.1 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 die Last auf den Web-Server durch den Einsatz von Content Delivery Networks reduzieren.

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.
- Content Delivery Networks fungieren als geografisch verteilte Caches statischer Inhalte.

Die Teilnehmer kennen verschiedene Skalierungsstrategien (Scale Up, Scale Out – horizontal sowie vertikal) und können die geeignete Skalierungsstrategie auswählen.

- Caching kann die Last auf den Server-Systemen reduzieren.
- Vertikale Skalierung durch leistungsfähigere Infrastruktur.
- Vertikale Skalierung dadurch, dass verschiedene Komponenten – etwa Web- und Applikationsserver – auf getrennten Infrastruktur-Komponenten verteilt werden.
- Horizontale Skalierung dadurch, dass die Komponenten dupliziert und die Last zwischen ihnen verteilt wird.
- Horizontale Skalierung dadurch, dass die Applikation in Teilapplikationen aufgespalten wird.
- Horizontale Skalierung dadurch, dass Requests nach bestimmten Charakteristiken – etwa dem authentisierten Benutzer oder dem Standort des Clients – auf verschiedene Infrastruktur-Komponenten verteilt werden.

6.2.2 Was sollen die Teilnehmer verstehen?

- 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 Authentisierungsschemata (beispielsweise von Form-based auf Basic-Auth.) umgeschrieben werden können.
- Die Teilnehmer sollten verstehen, welche 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 recht stark unterscheiden.

6.2.3 Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die verschiedenen Standard-Mechanismen, mit denen Web-Server um Applikationslogik erweitert werden können.
- Die Teilnehmer kennen die verschiedenen Ansätze zur Lastverteilung einer Web-Applikation und können ihre Vor- und Nachteile sowie ihre Auswirkungen auf Architektur und Design der Applikation erklären.
- Die Teilnehmer kennen verschiedene Leistungsmerkmale der im Internet üblichen Web-Server (Apache Httpd, Microsoft IIS, nginx ...) im Bezug auf Speicher- oder Thread-Verbrauch pro Anwender bzw. Client-Request.
- Die Teilnehmer kennen mit Graceful Degradation und Progressive Enhancement verschiedene Strategien, die Oberfläche barrierefrei zu gestalten und 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 verschiedene Techniken, zur Verbesserung der wahrnehmbaren Performance wie beispielsweise
    - Request-Reduktion durch inlining
    - Sprites und Data URIs
    - Minimizer für JavaScript und CSS
    - Etc.
- Die Teilnehmer kennen die Funktionsweise von Web-Firewalls und wissen gegen welche Arten von Angriffen diese schützen.

6.3 Referenzen

[Kopparapu 2002]

[ESI]

[Waldo+1994]

[Buschmann+1996]

[Brewer 2004]

[Daswani+2007]

[Stoneburner+2004]

[OWASP]

[HTML5-SSE]

[Websockets]

[Bayeux]

[WCAG 2008]

[BITV 2011]

[W3C-Int]
[Pritchett-2008]

7 Entwurf von Web-Architekturen

Dauer: 180 Min	Übungszeit: 120 Min

7.1 Begriffe und Konzepte

Datenmodellierung, Funktionale Zerlegung, Repräsentation.

Verteiltes System, CAP-Theorem, BASE, ACID, Sicherheit von Web-Applikationen, Authentisierung, Autorisierung, Barrierefreiheit/Accessibility, Internationalisierung/Lokalisierung,

HTML, CSS, JavaScript, Trennen von Inhalt, Präsentation und Verhalten, Graceful Degradation, Progressive Enhancement, unobtrusive JavaScript.

7.2 Lernziele

7.2.1 Was sollen die Teilnehmer können?

- Die Teilnehmer können die Muster Command-Query-Separation, CQS, und Command-Query-Responsibility-Separation, CQRS, gezielt nutzen, um sowohl funktionale als auch die nicht-funktionale Anforderungen gezielt zu erfüllen.
- Die Teilnehmer können gezielt Regeln vorgeben, um beim Einsatz von Model-View-Controller-Frameworks sicherzustellen, dass nur erwünschte Funktionalitäten in den Views implementiert werden.
- Die Teilnehmer können mit geeigneten Beschreibungsformaten wie (X)HTML, CSV, JSON, XML o. ä. Client-Programme versorgen.
- Die Teilnehmer können die Belange Repräsentation, Formatierung und Interaktion klar im Markup unterscheiden und beim Entwurf von Web-Anwendungen berücksichtigen.
- 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).
- Die Teilnehmer können Architekturmuster einsetzen, um bei konkurrierenden Qualitätszielen je nach Gewichtung ein gewünschtes Optimum zu erreichen.
- Die Teilnehmer können die Reaktionsfähigkeit des Systems durch geeigneten Einsatz von asynchroner Verarbeitung verbessern.

7.2.2 Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen die grundsätzliche Aussage des CAP-Theorems und können für die Abwägung zwischen Skalierbarkeit und Verfügbarkeit einen Kompromiss zwischen Konsistenz und Aktualität der Daten finden.
- Die Teilnehmer verstehen, wie Netzwerk-Fehler in verteilten Systemen Inkonsistenzen oder besondere Fehlerzustände hervorrufen können.
- Die Teilnehmer kennen das CAP-Theorem und verstehen, wie sich Architekturen für konsistente Systeme und Systeme mit Partitionstoleranz unterscheiden.
- Die Teilnehmer verstehen, wie Angreifer Sicherheitslücken mit Hilfe von HTML- oder 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 Transaktionskonzept BASE - Basically Available, Soft State, Eventual Consistency (grundsätzlich verfügbar, Übergangszustand, 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.

7.2.3 Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die Grundmechanismen, mit denen im eigenen Technologieportfolio SQL-Injection, HTML-Injection, Cross-Site-Scripting (XSS) und Client-Side-Request-Forgery (CSFR) 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 Cache-DB.
- 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)
    - Server Push
    - HTML5 Server Sent Events
    - Websockets
    - Bayeux (als Vertreter für Long Polling).
- 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 wissen, dass jede Anwendung, die aus öffentlichen Netzen erreichbar ist, angegriffen wird.
- 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.
    - Ressource-Frameworks bieten Abstraktionen für Ressourcen und Repräsentationen in einer RESTful Architektur.
    - Daten-getriebene Frameworks bilden zur Datenerfassungen Datenbank-Tabellen in der Oberfläche ab.

Die Teilnehmer kennen Authentisierungsmechanismen wie OpenId oder OAuth.

7.3 Referenzen

[Waldo+1994]

[Buschmann+1996]

[Brewer 2004]

[Daswani+2007]

[Stoneburner+2004]

[OWASP]

[HTML5-SSE]

[Websockets]

[Bayeux]

[WCAG 2008]

[BITV 2011]
[W3C-Int]
[Pritchett-2008]
[HTML-CSS]
[ECMA-262]
[HTML5]
[Olsson 2007]
[Evans 2004]
[Hohpe+2003]
[Nottingham 1998]
[Fowler 2011]
[Abbott-2010]

8 Qualität in Web-Architekturen

Dauer: 120 Min	Übungszeit: 60 Min

8.1 Begriffe und Konzepte

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

8.2 Lernziele

8.2.1 Was sollen die Teilnehmer können?

- Die Teilnehmer können die Grundlagen der Risikoanalyse und Bedrohungsmodellierung erläutern und anwenden.
- Die Teilnehmer können die Prinzipien für ein sicheres Software-Design anwenden:
    - Principle of Least Privilege: gebe 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 können erläutern und an Beispielen belegen, 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.

8.2.2 Was sollen die Teilnehmer verstehen?

- Die Teilnehmer verstehen, dass Nutzerverhalten im Internet sprunghaft über alle erwarteten Maße hinaus steigen kann.
- Die Teilnehmer verstehen, dass Back-End-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 Back-End führt.
- Die Teilnehmer verstehen, dass sich mit verschiedenen Persistenzstrategien (z. B. Relational, Dokumenten-orientiert, Spalten-orientiert, Key-Value-Store, Hierarchisch, Objektorientiert) 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.

8.2.3 Was sollen die Teilnehmer kennen?

- Die Teilnehmer kennen die wesentlichen Kennzahlen, die für den Betrieb eines Web-Servers relevant sind:
    - Verbrauchte CPU-Zyklen
    - Anzahl von Prozessen / Threads
    - RAM-Verbrauch
    - Anzahl offener Connections
    - Durchschnittliche Antwortzeit
    - Durchschnittliche Größe der Log-Einträge pro Request
    - Durchschnittliches Volumen an Daten, die für die Verarbeitung temporär erzeugt wird
    - Dauer die temporäre Daten vorgehalten werden.

9 Beispielarchitekturen

Dauer: 60 Min	Übungszeit: keine

Dieser Abschnitt ist nicht prüfungsrelevant.

9.1 Begriffe und Konzepte

Nicht zutreffend.

9.2 Lernziele

Innerhalb jeder akkreditierten Schulung muss mindestens ein Beispiel einer Software-Architektur für eine Web-Applikation vorgestellt, besprochen und bewertet werden.

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

9.2.1 Was sollen die Teilnehmer können?

Nicht zutreffend.

9.2.2 Was sollen die Teilnehmer verstehen?

Nicht zutreffend.

9.2.3 Was sollen die Teilnehmer kennen?

Nicht zutreffend.

9.3 Referenzen

Keine. Schulungsanbieter sind für die Auswahl und Beschreibung von Beispielen verantwortlich.

10 Quellen und Referenzen zu Web-Architektur

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

A

[Abbot+2010]

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

B

[Bayeux]

Bayeux Protokoll - http://svn.cometd.org/trunk/bayeux/bayeux.html

[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

[Prittchett 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