Curriculum for

Certified Professional for Software Architecture (CPSA)®
Advanced Level

Module:
WEB

Web Architectures

Version 1.4-EN, January 29, 2020

International Software Architecture Qualification Board

Table of Contents

Introduction: General Information on the iSAQB Advanced Level

What does an Advanced Level Module teach?

- The iSAQB Advanced Level offers modular training in three areas of competence with flexible academic approaches. It takes into account individual leanings and focuses.
- The certification is achieved by writing a term paper. Experts designated by the iSAQB perform the assessment and administer the oral examination.

What capabilities do graduates of the Advanced Level (CPSA-A) acquire?

CPSA-A graduates are capable of the following:

- Independent and method-based design of medium- to large-scale IT systems.
- Responsibility for technology and content of IT systems of medium to high criticality.
- Development, design and documentation of measures for achieving non-functional requirements. Support of development teams in implementation of these measures.
- Control and implementation of architecture-related communication in medium to large development teams.

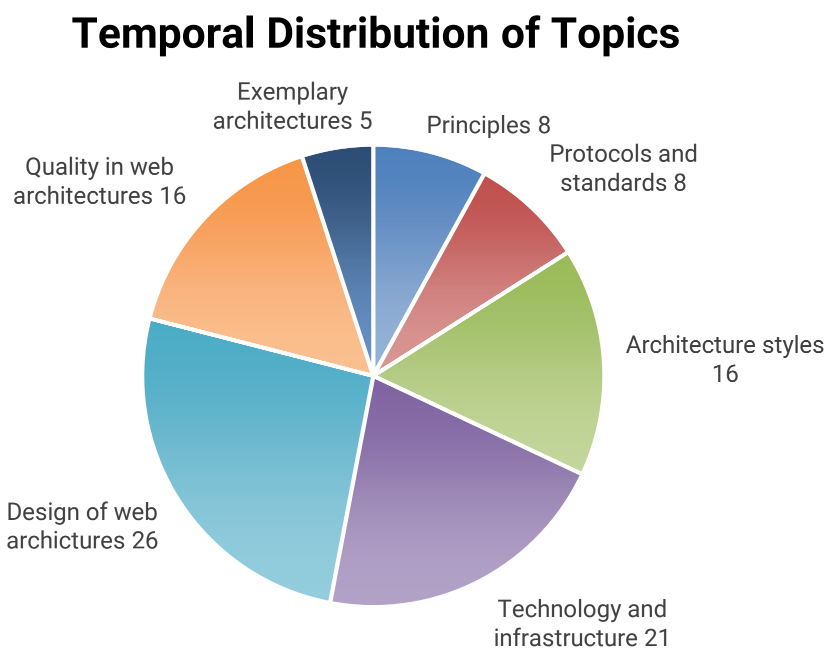
Requirements for CPSA-A certification

- Successful training and certification as a CPSA-F (Certified Professional for Software Architecture, Foundation Level)
- At least three years of full-time career experience in the IT sector, with participation in design and development of at least two different IT systems
  - Exceptions are possible on application (such as participation in open source projects)
- Training and advanced qualification within the framework of iSAQB Advanced Level courses comprising at least 70 credit points from all three different areas of competence (details in section 1.6).
  - Existing certifications can be accredited for these credit points on application. The list of current certificates accredited as credit points is available on the iSAQB website.
  - Other training and advanced qualifications can also be accredited on application to the iSAQB if they are relevant for software architecture. This will be decided on an individual basis by the iSAQB working group Advanced Level.
- Successful completion of the CPSA-A certification examination.

General Information about the Web Architecture Module

Outline of the curriculum for web architecture and recommended time plan

- Principles (at least 90 min)
- Protocols and standards (at least 90 min)
- Architecture styles (at least 180 min)
- Technology and infrastructure (at least 240 min)
- Design of web architectures (at least 300 min)
- Quality in web architectures (at least 180 min)
- Exemplary architectures (at least 60 min)



Duration, methodology and other details

The times stated below are recommendations. The minimum duration of a course on web architecture should be 3 days, but it can be longer. Providers can differ with respect to the duration, methodology, type and structure of the exercises as well as the detailed course outline. In particular, the curriculum leaves the type of examples and exercises completely open.

Requirements for the web architecture module

Participants **should** already have the following knowledge and/or experience:

- CPSA-F and all related prerequisites
- Experience with distributed systems – ideally web apps
- Basic knowledge in the web technologies HTML, CSS, JavaScript and at least one server-side framework

Outline of the web architecture curriculum

The single sections of the curriculum are described according to the following outline:

- **Terms/concepts:** Essential core concepts of this subject.
- **Instruction/exercise time:** Defines the minimum instruction and exercise time that is necessary for this subject or this exercise in an accredited course.
- **Learning goals:** Describes the content to be taught, including its core terms and concepts.

This section therefore also outlines the skills to be acquired in corresponding courses. The learning goals are differentiated in the following categories and sub-chapters:

- What should participants **be able to do**? Participants should be able to use this content independently after the course. This content is covered during the course by exercises or demonstrations and is part of the web architecture module examination and/or the final examination of the iSAQB Advanced Level.
- What should participants **understand**? This content can be tested in the web architecture module examination.
- What should participants **know**? This content (terms, concepts, methods, practices, etc.) can support understanding or motivate the subject. This content is not part of the examinations and will be mentioned in courses, but not necessarily taught in detail.
- **References:** References to secondary literature, standards or other sources. A detailed list of books and other sources is available on the iSAQB website under “Specialized sources”.

Supplementary information, terms and translations

If necessary for understanding of the curriculum, we have included technical terms in the iSAQB glossary, with definitions and, as needed, translations of the original literature.

Credit points for these courses

Courses licensed by the iSAQB in accordance with this curriculum contribute the following credit points for admission to the final Advanced Level certification examination:

Technical competence:	30 points

Introduction to the iSAQB Certification Program

Duration: 15 min (optional)	Practice time: none

This section is not relevant for the examination. This section can be omitted if participants are already CPSA-F certified.

Terms and concepts

iSAQB, Advanced Level certification and prerequisites for the same.

Learning goals

Participants become familiar with the iSAQB certification program and the corresponding examinations and examination procedures.

What should participants know?

- The iSAQB as an association
- Advanced Level as opposed to other levels
- Constraints and procedures of the iSAQB certification program

1 Fundamentals

Duration: 90 min	Practice time: none

1.1 Terms and concepts

Web browser, Web server, Client, Server, Proxy, Request, Response, Accessibility, Basic Auth, Intranet vs. Internet, Redirect, TLS, Prepocessor, Hypermedia, Statelessness, Pattern Libraries, Frontend Components, Ajax, Single-Sign-On (SSO), Separation of Concerns, Code on demand.

1.2 Learning goals

In this chapter, technology-independent concepts from the Web environment are discussed.

1.2.1 What should participants be able to do?

- Participants can explain the typical request/response process that takes place when an address is entered in the address line of the browser or when a form is sent.
- Participants can map the request/response process onto typical components in the web environment: client, server, proxy, reverse proxy, load balancer, DNS server, framework, own application logic (servlet, PHP script, Rails controller).
- Participants can explain the difference between a GET and POST request.
- Participants can make a diagram of the typical client-server infrastructure of web architectures.
- Participants can explain the general structure of HTTP requests: header (with host name, content type, etc.), content.
- Participants can explain the general structure of HTTP responses: response with status line, header and content.
- Participants can explain the structure of a URI and interpret the components in the request process that tend to be responsible for it.
- The participants can outline how a website with Ajax differs from the classic round-trip scheme without Ajax.
- The participants can explain the advantages of a stateless server regarding its scalability.
- The participants can split Web UIs into frontend components.

1.2.2 What should participants understand?

- Participants understand that the generic term web app can comprise fundamentally different types of IT systems with respect both to technology and content and that these systems require different architectures.
- Participants understand that web apps are not restricted to use by browsers, but can also be used by other clients.
- Participants understand the difference between structural and presentational characterization of content according to the Separation of Concerns.
- Participants understand the general requirements of the Web Content Accessibility Guidelines (WCAG), which are summarized under the term accessibility.
- Participants understand the flow of Single-Sign-On mechanisms (redirect, login, redirect with token, validation of the token), which is mostly used for web applications.
- Participants understand that a resource is more than just a database entity (e. g., also a process instance or a process step).
- Participants understand that hypermedia is used to connect resources to each other.
- Participants understand the concept of preprocessors for translating one language into another or for embedding dependencies or detecting dead code blocks.

1.2.3 What should participants know?

- Participants know that legal conditions can require barrier-free user interfaces. For example, the “Regulation on the creation of barrier-free information technology in accordance with the law on equal opportunities for the disabled” defines specific requirements that must be implemented by all websites of the Federal Administration of the Federal Republic of Germany.
- Some of the techniques that lead to barrier-free content also ensure that this content can be processed more easily by machine.
- Participants know different characteristics of web frameworks:
    - Component and request-response frameworks often use designs influenced by the MVC pattern.
    - Request-response frameworks are structurally mostly build in a relatively similar way: Actions accept the request, execute application logic and finally render the response, usually with templating (HTML) or object serialization (JSON/XML).
- Participants will be familiar with the concept of pattern libraries as a way of documenting front-end components.

2 Protocols and Standards

Duration: 90 min	Practice time: none

2.1 Terms and concepts

URI, URL, URN, HTTP/1.1, HTTP/2, HTTP/3, HTTP verbs (GET, PUT, POST, DELETE), HTTP headers, 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 Learning goals

2.2.1 What should participants be able to do?

- Participants can design software systems so that they use the protocols commonly used at the interfaces in the World Wide Web in an effective and resource-friendly manner.
- Participants can selectively use specific HTTP headers to enable the infrastructure to use caching.
- Participants can state which HTTP headers relate to caching and their effects (validation vs. time-to-live stamp).
- Participants can explain the difference between the roles that HTML, CSS and JavaScript play in the browser.

2.2.2 What should participants understand?

- The protocols and architecture of the web are not dependent on a particular technology.
- Participants understand the difference between server address and server name and know the effects for the generation and utilization of URLs, which should be used both system internally and externally.
- Participants understand the relationship between the transport protocol (TCP/UDP) and the application protocol (HTTP/1.1, HTTP/2 and HTTP/3 & QUIC).
- Participants understand the function of a name resolution via DNS lookup.
- Participants understand the effects of cache control headers on intermediaries.
- Participants understand the essential properties of the HTTP protocol and can explain them.
    - The HTTP protocol is a stateless request/response application protocol.
    - HTTP/1.1 is a text-based serial application protocol.
    - HTTP/2 multiplexes many requests/responses simultaneously over a TCP connection.
    - HTTP/3 exchanges TCP for UDP and complements the missing reliable connection in the form of QUIC on the application protocol level.
    - The participants know that HTTP/3, HTTP/2 and HTTP/1.1 implement largely the same semantics based on different line protocols.
    - The HTTP protocol expressly provides for intermediate processing processes (intermediaries).
    - The client identifies to the server a resource via a URI with the schema http or https. HTTP verbs define the type of access; the verbs have semantics.
    - The server responds with standardized status codes.
    - HTTP headers are used for additional services, metadata or extensions.
    - Various HTTP headers jointly define whether responses are allowed to be cached.

    - Different representations of the same resource are identified by media types (content types) and can be negotiated between the client and server by means of content negotiation.
    - Compression of the response data can (and should) also be negotiated via an analogous mechanism.
    - The HTTP protocol provides possibilities for different processes for authentication of the client to the server (e. g., basic authentication).
    - Cookies extend the protocol by a mechanism for annotating the browser session with any information.
    - Based on cookies, the inherently stateless protocol can be used by means of server-side sessions with status.
    - Participants understand the internal structure of HTTP requests and responses.
    - Participants understand that the protocol encrypts TLS at the transport level, during which symmetric encryption takes place.
    - The key is negotiated asymmetrically e. g., with certificates for the server and the client.
    - Client certificates can be used for authentication.
    - Certificates are managed using public key infrastructures (PKI).
- Participants understand that there are different authentication mechanisms for different requirements.
- Participants understand that the structure of information should be separated from its presentation.
- Participants understand how a redirect is processed.
- Participants understand the term idempotence.
- Participants understand the property of safety. They know that an HTTP GET is safe and that the server response to a GET can therefore be cached in principle.
- Participants understand why a browser requires a separate confirmation when "reloading" HTTP-POST.
- Participants understand how Web applications must be implemented to avoid such browser queries to a large extent.
- Participants understand what HTTPS means and how it works.
- Participants understand the effect of terminating a TLS connection.
- Participants know that with HTTP Basic Authentication, the user name and password are transferred in plain text.
- Participants know that confidential information in the Web environment is best transferred in encrypted form during transport. Transport level security (TLS) is used for encryption on the transport route.
- Participants understand the effects of TLS on caching.

2.2.3 What should participants know?

- Participants know the common HTTP status codes and know which cause is assumed and which response can normally be expected.
- Participants understand the responsibility and tasks of the protocols and components in the web environment:
    - URIs identify resources, URLs additionally localize them at an authority,
    - DNS servers provide support in the resolution of the authority part of the URI,
    - The HTTP protocol is a generic protocol for access to resources and provides solutions for several non-technical requirements,
    - Standardized formats are available for different types of data,
    - HTTP requires that clients and servers negotiate the format to be used if several alternatives exist.
- Participants know the browser-internal document object model and know that it can be changed by JavaScript and CSS.

- Participants know different data formats for the representation of information (HTML, XML, JSON, …).
- Participants know how cookies are exchanged and managed between the client and server.
- Participants know the XML Http Request Object (short XHR) as well as the HTML5-fetch API as basis of AJAX-based applications.
- Participants know the capabilities that result from server side events.
- Participants know mechanisms that allow the use of the browser buttons Back and Forward without triggering undesired side effects.
- Participants know the relevant standardization bodies such as IETF, IANA, W3C and their areas of responsibility with regard to web architecture.
- Participants know WebSockets.
- Participants know OAuth 2 as a system for delegating access to third parties.
- Participants know OpenID Connect as an OAuth 2 based system for Single-Sign-On.
- Participants know CORS as a mechanism for handling cross-domain restrictions.
- Participants know Content Security Policy as a mechanism to make cross-site scripting more difficult.

2.3 References

[RFC3986]

[RFC3987]

[RFC2616]

[Jacobs+2004]

[Fielding+2000]

[HTML-CSS]

[ECMA-262]

[WS-I]

[Tilkov 2011]

[RFC2246]

[RFC6265]

3 Architecture Styles

Duration: 120 min	Practice time: 60

3.1 Terms and concepts

Representational State Transfer (REST), Stateful Backend Web Apps, Single-page Application, Web Components.

3.2 Learning goals

3.2.1 What should participants be able to do?

- Participants can explain the difference between the REST style and the stateful backend architecture style.
- Participants can differentiate web architecture styles and choose which style makes the most sense for a quantity of given requirements and conditions.
- Participants can evaluate for a framework, which is also potentially unknown to them, how well it is suited for the realization of web applications according to the require-ments.
- Participants can explain different architecture styles of web architectures and design corresponding systems:
  - REST-compliant web apps
  - Stateful backend apps (e. g., component-oriented approaches).

3.2.2 What should participants understand?

- Participants understand the constraints that the architecture style REST has on the de-sign of a system:
  - Identifiable resources
  - Uniform interface
  - Stateless communication
  - Representations
  - Hypermedia.
- Participants understand the limitations of stateful backend architectures:
  - Communication takes place always against the same server instance.
  - In component-oriented approaches, the requests usually contain a command or a discriminator that decides which component is responsible for processing. Usually this dispatch information is hard-coded or stored in the main memory (in the session).
- Participants understand that no matter which architecture style is chosen, the core of ex-pertise, e. g., security and function-relevant tests or calculations, must always be imple-mented in the server.
- Participants understand how the response of web apps can be improved via functionality in the client i. e., via JavaScript in the client (e. g., in combination with AJAX)
- Participants understand the standard features of single page frameworks:
  - Databinding (two-way and one-way)
  - Templating
  - Client-side routing
  - Component abstraction.
- Participants understand that WebComponents (CustomElements + ShadowDOM) can be used to develop independent JavaScript components that are strongly isolated from the page and can be instantiated via DOM.

3.2.3 What should participants know?

- Participants know mechanisms to accept long-running transactions with corresponding status codes and to report the result via appropriate mechanisms as soon as it is available.
- Participants know the effects of mobile clients:
  - Widely differing bandwidths
  - Fluctuating, partly very high latency.
- Participants know that mobile clients sometimes have reduced performance with respect to main memory and CPU.
- Participants know that resources of mobile clients have to be conserved especially with regard to limited energy capacities of the end devices.

4 Technologies and Infrastructure

Duration: 180 min	Practice time: 60 min

4.1 Terms and concepts

Client, Server, Proxy, Reverse proxy, Content Delivery Networks/CDN, Load balancing, CGI, …

4.2 Learning goals

4.2.1 What should participants be able to do?

Participants can design server side web apps so that they effectively use the infrastructure.

Participants can improve the runtime behaviour of a system through the selective use of reverse proxies.

Participants can name different intermediaries and explain their effects on the architecture.

- Proxies save responses for the client; the HTTP protocol provides explicit rules for this.
- Reverse proxies are proxies on the server side and function as caches for the web app.
- Load balancers distribute requests on the server side to different servers.

Participants know different scaling strategies (horizontal and vertical) and can choose the suitable scaling strategy.

- Vertical scaling due to higher performance hardware.
- Vertical scaling due to the fact that different components – such as the web and app server – are distributed on separate infrastructure components.
- Vertical scaling through cluster solutions that simulate an application running on only one node even though there are multiple server nodes to run.
- Horizontal scaling by duplicating a server system and distributing the load as randomly and evenly as possible over the existing servers. For horizontal scaling, this method must be potentially infinite.

4.2.2 What should participants understand?

- Participants understand that caching can reduce the load on server systems.
- Participants understand how data from many web apps can be made available for few users (client or also forward proxy) or from many users for few web apps (reverse proxy).
- Participants understand how access to resources can be controlled by proxies.
- Participants understand how reverse proxy can be used to transcribe authentication schemas (for example from form-based to basic authentication).
- Participants should understand which infrastructure components in the request/response cycle affect the data throughput and latency time.
- Participants should understand how browser clients load resources and in particular how they evaluate HTML pages and query referenced resources.
- Participants understand how the transfer of resources to a content delivery network (CDN) helps to reduce the load on the internal infrastructure.
- Participants understand that not all web browsers support the standards to the same extent.
- Participants understand that the different web servers can differ greatly with respect to resource utilization.

4.2.3 What should participants know?

- Participants know the different standard mechanisms that can be used to expand web servers with application logic (CGI, Servlets, …).
- Participants know the different methods for load distribution of a web application (e. g., DNS-based load balancer or proxies).
- Participants know different strategies such as graceful degradation and progressive enhancement for the design of an interface adapted to the capabilities of the web browser.
- Participants know frameworks both for CSS and JavaScript that can facilitate development.
- Participants know that HTML5 includes diverse technology bundles that contain not only new structures for HTML, but also new APIs for JavaScript.
- Participants know the meaning of unobtrusive JavaScript: This comprises principles and practices that clearly separate content and behaviour.
- Participants know the functioning principle of web firewalls and know what types of attacks they provide protection against.

4.3 References

[Kopparapu 2002]
[ESI]
[Waldo+1994]

[Buschmann+1996]

[Brewer 2004]

[Daswani+2007]

[Stoneburner+2004]

[OWASP]

[HTML5-SSE]

[Websockets]

[WCAG 2008]

[BITV 2011]
[W3C-Int]
[Pritchett-2008]

5 Design of Web Architectures

Duration: 180 min	Practice time: 120 min

5.1 Terms and concepts

Data modelling, Functional decomposition, Representation.

Distributed system, CAP theorem, ACID, Security of web apps, Authentification, Authorization, Accessibility, Internationalization/Localization,

HTML, CSS, JavaScript, Separation of content, Presentation and behaviour, Graceful degradation, Progressive enhancement, Unobtrusive JavaScript.

5.2 Learning goals

5.2.1 What should participants be able to do?

- Participants can clearly differentiate the importance of representation, formatting and interaction in markup and take them into account in the design of web apps (separation of concerns).
- Participants can classify different tasks of the client interface formats from the existing web standards:
  - HTML provides the structure data
  - CSS defines the presentation. Selectors select elements from the HTML structure and assign them presentation characteristics.
  - JavaScript controls the behaviour, grants interactivity via event handlers and enables asynchronous server communication (AJAX).

5.2.2 What should participants understand?

- Participants understand the need for rules that ensure when using request-response frameworks that only desired functionality is implemented in the views/templates.
- Participants understand that the responsiveness of the system can often be improved by appropriate use of asynchronous processing.
- Participants understand the basic significance of the CAP theorem and know that a compromise must be found between scalability/availability/distribution and consistency/currentness.
- Participants understand how network errors in distributed systems can cause inconsistencies or special error states.
- Participants understand how architectures for consistent systems and systems with partition tolerance differ.
- Participants understand how attackers can exploit vulnerabilities using SQL injection, cross-site scripting (XSS) or client side request forgery (CSRF).
- Participants understand the transaction concept ACID – Atomicity, Consistency, Isolation and Durability.
- Participants understand the concept of Eventual Consistency (consistent after some delay).
- Participants understand that the inner architecture of the server side, in addition to the technical aspects, is also affected by
  - the philosophy of the framework used,
  - strategies for caching of static and dynamic content and
  - strategies for scaling the application.
- Participants understand that integration patterns such as messaging for a loose coupling of the web apps to background processing are suitable for long-running processes.

- Participants understand that any application accessible from public networks will be attacked.

5.2.3 What should participants know?

- Participants know the patterns of Command Query Separation, CQS, and Command Query Responsibility Separation, CQRS, in order to specifically meet both functional and non-functional requirements.
- Participants know that the consistency from "ACID" is a different consistency than the consistency from CAP.
- Participants know the basic mechanisms used to prevent SQL injection, cross site scripting (XSS) and client side request forgery (CSRF) in their own technology portfolio.
- Participants know various strategies for management of session-related data, e.g. in the cookie, in the main memory of the server, in a database, to comply with quality requirements.
- Participants know the difference between internationalization and localization.
- Participants know different strategies with which a client can be informed of the result of an activity that was started asynchronously on the server. For example:
    - Polling (client pull)
    - HTML5 server sent events
    - Web sockets.
- Participants know that many server side frameworks allow plug-ins, analogous to the pipes and filter pattern, for example servlet filters in Java Servlet API based frameworks or HttpModules in ASP.NET.
- Participants know at least one server side framework that can be used to implement web apps.
- Participants know different types of server side web frameworks:
    - Component frameworks attempt to transfer an event-based programming model from the area of desktop applications.
    - Action or request response frameworks represent the HTTP protocol directly.
    - Data-driven frameworks represent database tables in the interface for data input.

Participants know authentication mechanisms such as OpenId Connect or CAS.

5.3 References

[Waldo+1994]

[Buschmann+1996]

[Brewer 2004]

[Daswani+2007]

[Stoneburner+2004]

[OWASP]

[HTML5-SSE]

[Websockets]

[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]

6 Quality in Web Architectures

Duration: 120 min	Practice time: 60 min

6.1 Terms and concepts

Security, Scalability, Web scale, Availability, Operability, Accessibility.

6.2 Learning goals

6.2.1 What should participants be able to do?

6.2.2 What should participants understand?

- Participants understand the basics of risk analysis and threat modelling.
- Participants understand the principles of secure software design:
  - Principle of Least Privilege: give users and components only as many rights as they absolutely need.
  - Defense in Depth: do not trust that other security measures have worked - use redundant checks.
  - Secure by Default: rights must be granted explicitly.
  - Don't trust anybody: user input and data from external sources must be validated.
  - Compartmentalize: Separate components that require different rights from each other.
  - Fail securely: If errors occur, no security-relevant information may be disclosed.
- Participants understand that the number of accesses to the Internet can increase by leaps and bounds beyond all expected levels.
- Participants understand that backend systems (such as databases or enterprise information systems) can easily become overloaded if the number of requests increases because a request frequently leads to numerous queries to the backend.
- Participants understand that different persistence strategies (e. g., relational, aggregation/document-oriented, column-oriented, hierarchical, object-oriented, graph-oriented) can be used to support different utilization scenarios with varying degrees of effectiveness.
- Participants understand that the quality depends both on the suitability of the chosen technology and on the specific implementation.
- Participants understand that internationalization means more than just providing content in different languages. For instance:
  - Formats for numbers or dates or the order in which words are sorted depend on the cultural area.
  - Font directions may differ.
  - Layouts must take into account that texts in different languages take up different amounts of space.
  - The legal framework in different countries must be taken into account.
  - Cultural differences affect, for example, the choice of colors.

6.2.3 What should participants know?

- Participants know the essential performance indicators that are relevant for operation of a web application:
  - Number of processes/threads
  - RAM utilization

  - Average resource consumption per request
  - Number of open connections
  - Average response time

7 Exemplary Architectures

Duration: 60 min	Practice time: none

This section is not relevant for the examination.

7.1 Terms and concepts

Not applicable.

7.2 Learning goals

Within each accredited course specific examples of different areas of web architecture must be presented, discussed and evaluated:

- Dealing with infrastructure and especially reverse proxies.
- A rough pattern of a web backend application.
- One or more examples of single page framework features such as data binding or client side routing.
- Examples of different database models.

The type and character of the examples presented can depend on the course and the interests of the participants and are not prescribed by the iSAQB.

7.2.1 What should participants be able to do?

Not applicable.

7.2.2 What should participants understand?

Not applicable.

7.2.3 What should participants know?

Not applicable.

7.3 References

None. Training course providers are responsible for the selection and description of examples.

Sources and References on Web Architecture

This section contains references that are referred to in whole or in part in the curriculum.

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

[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