Curriculum for

CPSA Certified Professional for Software Architecture®

– Advanced Level –

# Module: WEB

# Web Architectures

International Software Architecture Qualification Board

Version 1.3 (February 2015)

## Table of contents

# 0 Introduction: General Information on the iSAQB Advanced Level

## 0.1 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.

## 0.2 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.

## 0.3 Requirements for CPSA-A certification

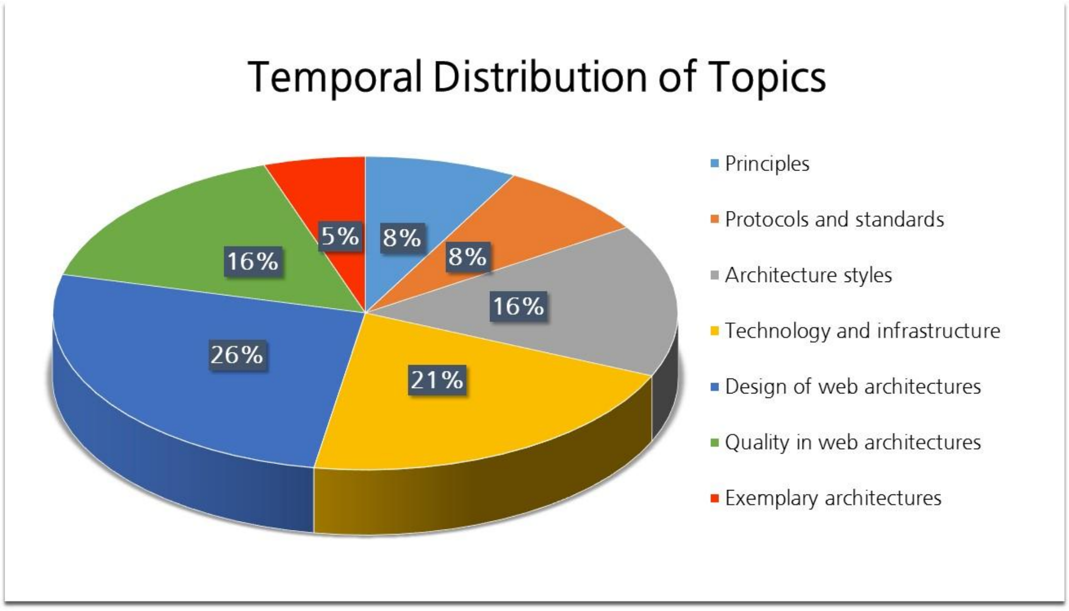
- Successful training and certification as a CPSA-F (Certified Professional for Software Architecture, Foundation Level)
- At least three years of full-time career experience in the IT sector, with participation in design and development of at least two different IT systems
    - Exceptions are possible 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.

CPSA-A®
Certified Professional for Software Architecture
Advanced Level

# 1 General Information about the Web Architecture Module

## 1.1 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)



## 1.2 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.

## 1.3 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

## 1.4 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 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".

## 1.5 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.

## 1.6 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 |

# 2 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.

## 2.1 Terms and concepts

iSAQB, Advanced Level certification and prerequisites for the same.

## 2.2 Learning goals

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

### 2.2.1 What should participants know?

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

# 3 Fundamentals

| Duration: 90 min | Practice time: none |
|---|---|

## 3.1 Terms and concepts

Web browser, Web server, Client, Server, Proxy, HTTP, HTML, CSS, JavaScript, URI, Request, Response, Accessibility, Basic Auth, Digest Auth, Intranet vs. Internet, Redirect, TLS, SSL.

## 3.2 Learning goals

### 3.2.1 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 components (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.
- Participants know what roles HTML, CSS and JavaScript play in the browser.

### 3.2.2 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 implement both online transaction and online analysis applications, for which different architecture patterns are used.
- Participants understand that web apps are not restricted to use by browsers, but can also be used by other clients.
- Participants understand how a redirect is processed.
- Participants understand why a browser requests a separate confirmation during “Reload” in an HTTP-POST.
- Participants understand how web apps have to be implemented to prevent such return queries by the browser.
- Participants understand what HTTPS means and how it works.
- Participants understand the effect of terminating an SSL connection.
- Participants understand the difference between structural and presentational characterization of content.
- Some of the techniques that result in barrier-free content also facilitate automatic processing of the content.

### 3.2.3 What should participants know?

- Participants are familiar with the relevant standardization committees such as IETF, IANA, W3C and their responsibilities with respect to web architecture.
- 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.
- Participants know the general requirements included in the term accessibility.
- Component as well as action or request/response frameworks frequently use designs that are influenced by the MVC pattern.
- Participants know at least one client technology/one markup language to convey information to the client and therefore to implement the desired functionality.
- Participants know that HTTP is a test-based application protocol.
- Participants know the different standardized authentication mechanisms: HTTP Basic and Digest authentication.
- Participants know that the user name and password are transmitted in plain text in HTTP Basic authentication.
- Participants know that confidential information in the web environment should preferably be in encrypted form during transport. Encryption during transport (Transport Level Security, TLS) takes place by means of SSL.
- Participants are familiar with the effects of SSL for caching.
- Participants are familiar with web sockets and SPDY.

# 4 Protocols and Standards

| Duration: 90 min | Practice time: none |
|---|---|

## 4.1 Terms and concepts

URI, URL, URN and IRI, HTTP, HTTP verbs (GET, PUT, POST, DELETE), HTTP headers, Intermediaries, Caching, Idempotence, Content types, Content negotiation, SSL/TLS, HTML, DOM, RSS, ATOM.

## 4.2 Learning goals

### 4.2.1 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 state what kind of queries lead to which data traffic.
- Participants can selectively use specific HTTP headers to enable the infrastructure to use caching.
- Participants can use suitable authentication mechanisms (basic and digest authentication) for given requirements.
- Participants can design web apps in which the Forward and Back browser buttons generally function without undesired side effects.
- Participants can secure the communication channel with TLS and know which authentication mechanism is suitable in the event of unencrypted communication.
- Participants can estimate the size of dynamic content for standard formats (both in the browser and during transport).
- Participants can state which HTTP headers relate to caching and their effects (validation vs. time-to-live stamp).
- Participants can distinguish between the structure and presentation of information.
- Participants can classify the size relationships of payload and overhead for the formats and can choose the suitable form of representation based on the expected data traffic and the planned utilization.

### 4.2.2 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/IP) and the application protocol (HTTP).
- 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 text-based and stateless request/response application protocol.
    - 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.
    - HTTP verbs, status codes and diverse 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.
  - The HTTP protocol contains a sub-protocol for authentication of the client to the server (Basic and Digest authentication).
  - Cookies expand the stateless protocol to include a mechanism for stateful communication.
  - Participants understand the internal structure of HTTP requests and responses.
  - Participants understand that the SSL protocol encrypts at the transport level (TLS), during which symmetric encryption takes place.
  - The key is negotiated asymmetrically with certificates for the server and the client.
  - Client certificates can be used for authentication.

### 4.2.3 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 and localize resources,
  - 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,
  - Encryption takes place beneath the level of the HTTP protocol,
  - 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 that is used and can use it to evaluate or manipulate documents in the browser.
- Participants know different data formats for the representation of information (HTML, XML, XHTML, JSON, CSV…).
- Participants know the term idempotence and know for which HTTP verbs requests are idempotent and for which the response of the server is allowed to be cached.
- Participants know that web servers are frequently sub-divided internally into components for implementation of the HTTP server and for implementation of the application codes (for example web containers and application containers).
- Participants know how cookies are exchanged and managed between the client and server.
- Participants know different formats for implementing feeds such as RSS and Atom.
- Participants know the XMLHttpRequest object (XHR for short) as the basis of AJAX-based apps.
- Participants know the capabilities that result from server side events.

## 4.3 References

[RFC3986]

[RFC3987]

[RFC2616]

[Jacobs+2004]

[Fielding+2000]

[HTML-CSS]

[ECMA-262]

[WS-I]

[Tilkov 2011]

[RFC2246]
[RFC6265]

# 5 Architecture Styles

| Duration: 120 min | Practice time: 60 |
|---|---|

## 5.1 Terms and concepts

Representational State Transfer (REST), Resource-oriented Client Architecture (ROCA), Single URI/stateful backend web apps, Single-page application, Cross-compiler architectures, Fat client in browser apps, Smart client in browser apps.

## 5.2 Learning goals

### 5.2.1 What should participants be able to do?

- Participants can explain the difference between the REST style and the single URI/stateful backend architecture style.
- Participants can differentiate architecture styles and choose which style makes the most sense for a quantity of given requirements and conditions.
- Participants can design interfaces so that long-running transactions are accepted with corresponding status codes and that the result is reported by means of suitable mechanisms as soon as it is available.
- Participants can assess how good a given framework is for implementing web apps in accordance with a specified architecture style.
- Participants can explain different architecture styles of web architectures and design corresponding systems:
  - REST-compliant web apps
  - Single-URI/Single-method apps
  - Fat client in browser apps
  - Smart client in browser apps.

### 5.2.2 What should participants understand?

- Participants understand the constraints that the architecture style REST has on the design of a system:
  - Identifiable resources
  - Uniform interface
  - Stateless communication
  - Representations
  - Hypermedia.
- Participants understand the limitations of single URI/stateful backend architectures:
  - Communication takes place via a central dispatcher.
  - Requests contain a command or a discriminator that decides which component is responsible for the processing.
  - This dispatch information is generally hard coded or stored in main memory.
- Participants can weigh the advantages and disadvantages in the use of tools for generating client-specific artifacts such as HTML/JavaScript, etc. from server side source code.
- Participants understand that no matter which architecture style is chosen, security-related checks always have to be implemented in the server.
- Participants understand how the response of web apps can be improved via functionality in the client. For example:
  - JavaScript in client (e.g. in combination with AJAX)
  - Proprietary extensions such as Adobe Flash or Microsoft Silverlight
  - Smart client apps.

### 5.2.3 What should participants know?

- Participants know the effects of mobile clients:
    - Widely differing bandwidths
    - Partly very low bandwidths
    - Fluctuating bandwidths.
- Participants know that mobile clients sometimes have reduced performance with respect to main memory and CPU.

# 6 Technologies and Infrastructure

| Duration: 180 min | Practice time: 60 min |
|---|---|

## 6.1 Terms and concepts

Client, Server, Proxy, Reverse proxy, Content Delivery Networks/CDN, Load balancing, CGI, FastCGI, Servlets, ActiveServerPages, PHP pages…

## 6.2 Learning goals

### 6.2.1 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 reduce the load on web servers by using content delivery networks.

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.
- Content delivery networks function as geographically distributed caches of static content.

Participants know different scaling strategies (scale up, scale out – horizontal and vertical) and can choose the suitable scaling strategy.

- Caching can reduce the load on the server systems.
- Vertical scaling due to higher performance infrastructure.
- Vertical scaling due to the fact that different components – such as the web and app server – are distributed on separate infrastructure components.
- Horizontal scaling due to the fact that components are duplicated and the load is distributed between them.
- Horizontal scaling due to the fact that the application is divided into sub-applications.
- Horizontal scaling due to the fact that requests are distributed to different infrastructure components based on certain characteristics – such as the authenticated user or the location of the client.

### 6.2.2 What should participants understand?

- 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 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.

### 6.2.3 What should participants know?

- Participants know the different standard mechanisms that can be used to expand web servers with application logic.
- Participants know the different methods for load distribution of a web application and can explain their advantages and disadvantages as well as their effects on the architecture and design of the app.
- Participants know different performance characteristics of the standard web servers in the Internet (Apache Httpd, Microsoft IIS, nginx, etc.) with respect to memory or thread utilization per user or client request.
- Participants know different strategies such as graceful degradation and progressive enhancement for the design of a barrier-free interface and adaptation 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 different techniques for improving the perceivable performance, such as
  - Request reduction through inlining
  - Sprites and data URIs
  - Minimizers for JavaScript and CSS
  - Etc.
- Participants know the functioning principle of web firewalls and know what types of attacks they provide protection against.

## 6.3 References

[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 Design of Web Architectures

| Duration: 180 min | Practice time: 120 min |
|---|---|

## 7.1 Terms and concepts

Data modelling, Functional decomposition, Representation.

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

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

## 7.2 Learning goals

### 7.2.1 What should participants be able to do?

- Participants can selectively use the patterns Command Query Separation (CQS) and Command Query Responsibility Separation (CQRS) to selectively fulfil both functional and non-functional requirements.
- Participants can selectively specify rules to ensure that only desired functionalities are implemented in the views when model view controller frameworks are used.
- Participants can use suitable description formats such as (X)HTML, CSV, JSON, and XML to supply client programs.
- Participants can clearly differentiate the importance of representation, formatting and interaction in markup and take them into account in the design of web apps.
- 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).
- Participants can use architectural patterns to achieve a desired optimum among concurring quality goals, depending on the weighting.
- Participants can improve the response of the system through the suitable use of asynchronous processing.

### 7.2.2 What should participants understand?

- Participants understand the basic significance of the CAP theorem and can find a compromise between consistency and timeliness of data in order to choose between scalability and availability.
- Participants understand how network errors in distributed systems can cause inconsistencies or special error states.
- Participants know the CAP theorem and understand the difference between architectures for consistent systems and systems with partition tolerance.
- Participants understand how attackers use security gaps with the help of HTML or 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 transaction concept BASE – Basically Available, Soft State, Eventual Consistency.

- 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.

### 7.2.3 What should participants know?

- Participants know the basic mechanisms used to prevent SQL injection, HTML injection, cross site scripting (XSS) and client side request forgery (CSFR) 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 cache DB, 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)
  - Server push
  - HTML5 server sent events
  - Web sockets
  - Bayeux (as a method of long polling).
- 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 Http-Modules in ASP.NET.
- Participants know that every application that is accessible from public networks can be attacked.
- 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.
  - Resource frameworks offer abstractions for resources and representations in a RESTful architecture.
  - Data-driven frameworks represent database tables in the interface for data input.

Participants know authentication mechanisms such as OpenId or OAuth.

## 7.3 References

[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 Quality in Web Architectures

| Duration: 120 min | Practice time: 60 min |
|---|---|

## 8.1 Terms and concepts

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

## 8.2 Learning goals

### 8.2.1 What should participants be able to do?

- Participants can explain and apply the fundamentals of risk analysis and threat modelling.
- Participants can apply the principles for a secure software design:
    - Principle of least privilege: give users and components only as many rights as they absolutely need.
    - Defence in depth: do not rely on other security measures – 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 need different rights.
    - Fail securely: If errors occur, no security-related information must be exposed.
- Participants can explain and demonstrate with examples that internationalization means more than offering content in different languages. For example:
    - Formats for numbers or dates or the sequence in which words are sorted depends on the cultural environment.
    - Text directions can differ.
    - Layouts must take into account that texts in different languages need different amounts of space.
    - The legal requirements in different countries must be taken into account.
    - Cultural differences affect the choice of colours, for example.

### 8.2.2 What should participants understand?

- Participants understand that user behaviour in the Internet can rapidly exceed all expected volumes.
- 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, document-oriented, column-oriented, key value store, hierarchical, object-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.

### 8.2.3 What should participants know?

- Participants know the essential performance indicators that are relevant for operation of a web server:
    - Used CPU cycles
    - Number of processes / threads
    - RAM utilization
    - Number of open connections

- Average response time
- Average size of log entries per request
- Average volume of data temporarily generated for processing
- Duration that temporary data is retained.

# 9 Exemplary Architectures

| Duration: 60 min | Practice time: none |
|---|---|

This section is not relevant for the examination.

## 9.1 Terms and concepts

Not applicable.

## 9.2 Learning goals

Within each accredited course at least one example of a software architecture for a web app must be presented, discussed and evaluated.

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.

### 9.2.1 What should participants be able to do?

Not applicable.

### 9.2.2 What should participants understand?

Not applicable.

### 9.2.3 What should participants know?

Not applicable.

## 9.3 References

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

## 10 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

[Bayeux]

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

[BITV 2011]

Verordnung zur Schaffung barrierefreier Informationstechnik nach dem Behindertengleichstellungs-gesetz (Barrierefreie-Informationstechnik-Verordnung - BITV 2.0) – http://www.gesetze-im-inter-net.de/bitv_2_0/BJNR184300011.html

[Brewer 2004]

Brewer, E.: Towards Robust Distributed Systems. Keynote zur PODC (Symposium on Principles of Dis-tributed 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