

Webengineering und Einsatz von Webtechnologien

Robin Rausch, Florian Maslowski, Ozan Akzebe

18. Mai 2023

Inhaltsverzeichnis

1	Architektur verteilter Anwendungen	4
2	Application programming interface - API	4
3	Das Web	5
3.1	HTML, CSS und Forms	5
3.1.1	HTML	5
3.1.2	CSS	5
3.1.3	Forms	5
4	Internet	5
4.1	Intranet	5
4.2	Network Address Translation - NAT	5
4.2.1	Source-NAT	6
4.2.2	Destination-NAT	6
5	TCP/IP Grundlagen	6
5.1	OSI 7 Schichtenmodell	6
5.2	IP	6
5.2.1	IPv4	6
5.2.2	IPv6	6
5.2.3	Routing in IP	7
5.2.4	Subnetze	7
5.2.5	IP in MAC übersetzen - ARP	7
5.2.6	Adress Resolution Protocol	7
5.2.7	Reverse Adress Resolution Protocol (RARP)	7
5.2.8	ICMP	7
5.2.9	Kommunikation zwischen IPv4 und IPv6	8
5.3	Transmission Control Protocol - TCP	8
6	Domain Name System - DNS	8
6.1	Einordnung der Begriffe	8
6.2	DNS Zonen	9
6.3	DNS Server	9
6.4	DNS Records	10

7	Webserver	10
7.1	SSL/TLS Verschlüsselung	10
7.1.1	Symmetrische Verschlüsselung	10
7.1.2	Asymmetrische Verschlüsselung	10
7.1.3	Public-Key Infrastruktur	11
7.1.4	TLS Handshake	11
8	HTTP	12
8.1	Die Rollen in HTTP	13
8.2	Persistente Verbindungen	13
8.3	Chunked Transfer	13
8.4	URL	13
8.5	HTTP Caching	13
8.6	HTTP Cookies	13
9	Kommunikation	14
9.1	Polling	14
9.2	Long Polling	14
9.3	Server-Sent Events - SSE	15
9.4	WebSockets	16
10	Webformate	17
10.1	CSV	17
10.2	XML (Extensible Markup Language)	17
10.2.1	XML Schema	17
10.3	JSON	18
11	Serverseitige Programmierung	18
12	Node.JS	19
13	Web Services	19
13.1	SOAP	20
13.1.1	SOAP Nachrichten	21
13.1.2	SOAP - WSDL	22
13.1.3	UDDI	23
13.2	REST	23
13.3	GraphQL	25
14	Message Queueing	25
14.1	Message Queue Manager - MQM	26
15	Datenbanken	26
15.1	Relationale Datenbanken	26
15.2	Nicht-Relationale Datenbanken - NoSQL	26
15.2.1	Nicht-Relationale Key-Value Datenbanken	27
15.2.2	Nicht-Relationale Dokumentenorientierte Datenbanken	27
15.2.3	Nicht-Relationale Graphendatenbanken	28
15.2.4	Nicht-Relationale Spaltendatenbanken	28

16 Testing	28
16.1 Test Driven Development - TDD	29
16.2 Testabdeckung	29
16.3 Test-Doubles	30
17 Deployment and Hosting	30
18 Sicherheit	31
18.1 Injeciton	31
18.2 Broken Authentication	31
18.3 Sensitive Data Exposure	31
18.4 XML External Entities	32
18.5 Broken Access Control	32
18.6 Security Misconfiguration	32
18.7 Cross-Site Scripting	32
18.8 Insecure Deserialization	32
18.9 Using Components with Known Vulnerabilities	33
18.10 Unsufficient Logging & Monitoring	33

1 Architektur verteilter Anwendungen

Aufteilung in:

- Präsentationsschicht
 - Schnittstelle zwischen Benutzer und Anwendung
 - Rendern von Daten
 - Input/Output und Kommunikation
- Anwendungsschicht
 - Funktionen welcher der Präsentationsschicht zur Verfügung gestellt werden
 - Anwendungslogik, Abläufe und eigentlicher Sinn und Zweck der Anwendung
- Datenschicht
 - Datenbanken
 - Dateien
 - Warteschlangen
 - APIs

Strukturen sind:

- 1-Tier Structure
 - Alles beim Client
- 2-Tier Structure
 - Client-Server Struktur
 - Schwierigkeit: Entscheiden was bei wem läuft
- 3-Tier Structure
 - besitzt in der Regel eine Middleware für die Interaktionslogik
 - Client greift auf Middleware zu
 - Aus Sicht des Clients ist die Middleware die Applikationslogik
- N-Tier Structure
 - Moderne Anwendungen sind häufig sehr stark vernetzt
 - Zugriff auf unterschiedliche Anwendungen die wiederum auf unterschiedlichen Architekturen basieren

2 Application programming interface - API

APIs sind Schnittstellen, welche Funktionen für Clients ähnlicher Programme oder Anwendungen zugänglich macht. Diese kann sowohl lokal, als auch remote sein.

3 Das Web

Hypermedial: Nicht lineare Form von Medien, die durch Hyperlinks miteinander verknüpft sind.

Ein Link hat zwei Aufgaben:

- Es muss ein Objekt eindeutig identifizieren **wo** es gefunden werden kann
- Es muss wissen, **wie** es gefunden werden kann

3.1 HTML, CSS und Forms

3.1.1 HTML

Hypertext Markup Language ist keine Programmiersprache sondern eine Markup Language! HTML Code besteht aus Tags(z.B. <h1>TITEL</h1>). Das gesamte HTML Dokument ist in <html>-tags eingeschlossen.

3.1.2 CSS

Cascading Stylesheets überschreibt die Richtlinien wie der HTML Code dargestellt wird. Es gibt drei Möglichkeiten CSS in HTML einzubinden:

- Referenz auf externe CSS Datei
- Style Definition innerhalb des headers
- Style Definition innerhalb der Tags

3.1.3 Forms

Formulare geben die ermöglichen den Endnutzern eine strukturierte Eingabe von Informationen. Diese Eingaben können dann an ein spezifiziertes Programm gesendet werden. Formulare sind die Grundlage für die Interaktion mit Benutzern.

4 Internet

Das Internet basiert auf Client-Server Strukturen. Anfragen gehen immer vom Client aus. Technische Basis des Internets ist das Internet Protocol(IP). Dadurch ist jeder Rechner eindeutig identifizierbar. Typische Dienste sind Mailserver, Fileserver oder Datenbankserver.

4.1 Intranet

Das Intranet ist das heimische Netz und endet am Router. Nicht jeder Rechner im Intranet ist von außerhalb ansprechbar.

4.2 Network Address Translation - NAT

Dient zum Verbinden von unterschiedlichen Netzen, zwischen denen kein Routing stattfinden kann. Wir unterscheiden: Source-NAT und Destination-NAT.

4.2.1 Source-NAT

Mehrere Geräte im Netzwerk kommunizieren über gemeinsame IP-Adresse nach außen. Der Gateway ersetzt die interne Adresse mit der öffentlichen Adresse und ändert die in der Datenpaketen stehenden Quell- und Zieladressen, damit das Paket das Ziel erreicht. Bei zurückkehrenden Paketen erfolgt umgekehrte Ersetzung. Der Gateway speichert dazu die Anfragen in einer NAT-Tabelle damit Zuordnung erfolgen kann.

4.2.2 Destination-NAT

Wird verwendet um die Zieladresse der eingehende Pakete zu ändern. Anfrage an 74.123.23.48:80 wird bspw. Zu 192.168.10.10:80 ersetzt. Wird benötigt um aus dem Internet auf Geräte innerhalb eines vom Internet abgetrennten Netzes zuzugreifen. (Oftmals auch „Portfreigabe“ genannt)

5 TCP/IP Grundlagen

5.1 OSI 7 Schichtenmodell



5.2 IP

Das Internet-protocol ist ein Protokoll um Datenpakete zu übermitteln und optimiert für das Zerlegen, Routen und Zusammensetzen von Datenpaketen. Es erlaubt es Server und Clients weltweit eindeutig identifizieren zu können. Es bildet IP-Adressen auf MAC und ARP ab. Außerdem handhabt es die Fragmentierung von Paketen.

5.2.1 IPv4

32 bit Adressen mit 4 je 8 bit Gruppen. Server haben eine feste IP-Adresse und Clients haben eine temporäre. Diese werden durch NICs (Network Information Center) zugewiesen. Trennung durch einfachen Punkt.

5.2.2 IPv6

128 bit Adresse mit 8 Gruppen zu 16 bit. Trennung durch Doppelpunkt.

5.2.3 Routing in IP

Bei Erhalt eines IP Pakets bestimme ob die Zieladresse die eigene Adresse ist. Wenn Ja, dann gebe in nächste Ebene. Wenn nein, dann prüfe ob Ziel im eigenen Netz ist. Ist diese im eigenen Netz, dann sende direkt an Ziel. Ansonsten führe Pfadselektion durch: transferiere Paket zu nächstem Knoten spezifiziert durch Subnetzmaske. Wenn kein Eintrag gefunden wird, gebe das Paket an den „Standard Router“, definiert im Routing Table. Hierzu existieren Routing Tabellen. Um IPv4 Adressen besser einordnen zu können gibt es Netzklassen (Klasse A ist bsp. für große Organisationen, da diese viele IP Adressen brauchen).

5.2.4 Subnetze

Als Subnetz wird ein Teilnetz eines Netzes beim Internetprotokoll bezeichnet. Es fasst mehrere aufeinanderfolgende IP-Adressen mittels einer Subnetzmaske an binären Grenzen unter einem gemeinsamen Vorderteil, dem Präfix zusammen.

5.2.5 IP in MAC übersetzen - ARP

Dafür gibt es 3 Möglichkeiten:

- Jeder Server hält eine Tabelle
 - Pro: Übersetzung schnell und einfach
 - Contra: Hoher Managementaufwand
- Zentraler Server hält eine Tabelle für alle Rechner in lokalem Netz
 - Pro: Reduzierter Management Aufwand
 - Contra: Server muss hochverfügbar sein
- Dynamisches erzeugen (und pflegen) der Tabellen mit Hilfe von Broadcasts
 - Standard Technik fürs Internet → ARP

5.2.6 Adress Resolution Protocol

Client prüft in lokaler Tabelle ob ein Eintrag vorhanden ist für IP zu MAC. Wenn nicht vorhanden, sendet der Client eine Nachricht (eigene IP, eigene MAC, unbekannte IP) an ALLE lokalen Hosts. Der Host mit der „unbekannten IP“ antwortet mit seiner MAC Adresse. Alle anderen Empfänger machen nichts. Client aktualisiert seinen Table.

5.2.7 Reverse Adress Resolution Protocol (RARP)

Nicht jeder Computer hat eine feste IP Adresse. Die Mac Adresse hingegen ist persistent auf der Netzwerkkarte. Client sendet seine Mac Adresse mit einem RARP Broadcast in lokales Netzwerk. RARP Server weißt eine IP Adresse zu und sendet diese zurück.

5.2.8 ICMP

ICMP wird verwendet um eine Kontrollnachricht zu senden. Gibt Fehlermeldungen und Informationen zurück.

5.2.9 Kommunikation zwischen IPv4 und IPv6

Dual Stack beschreibt den Parallelbetrieb von IPv4 und IPv6 an einem Internetanschluss. Dabei wird dem Zugang eine öffentliche IPv4 Adresse und ein IPv6 Netz bereitgestellt. Geräte im Netzwerk müssen unterstützen, dass IPv4 und IPv6 parallel kommuniziert wird. Problem: Viele alte Geräte unterstützen das noch nicht.

Beim Dual Stack Lite wird eine IPv4 Adresse durch die Übersetzung beim NAT an mehrere IPv6 Adressen verteilt. Somit teilen sich mehrere IPv6 Adressen eine IPv4 Adresse.

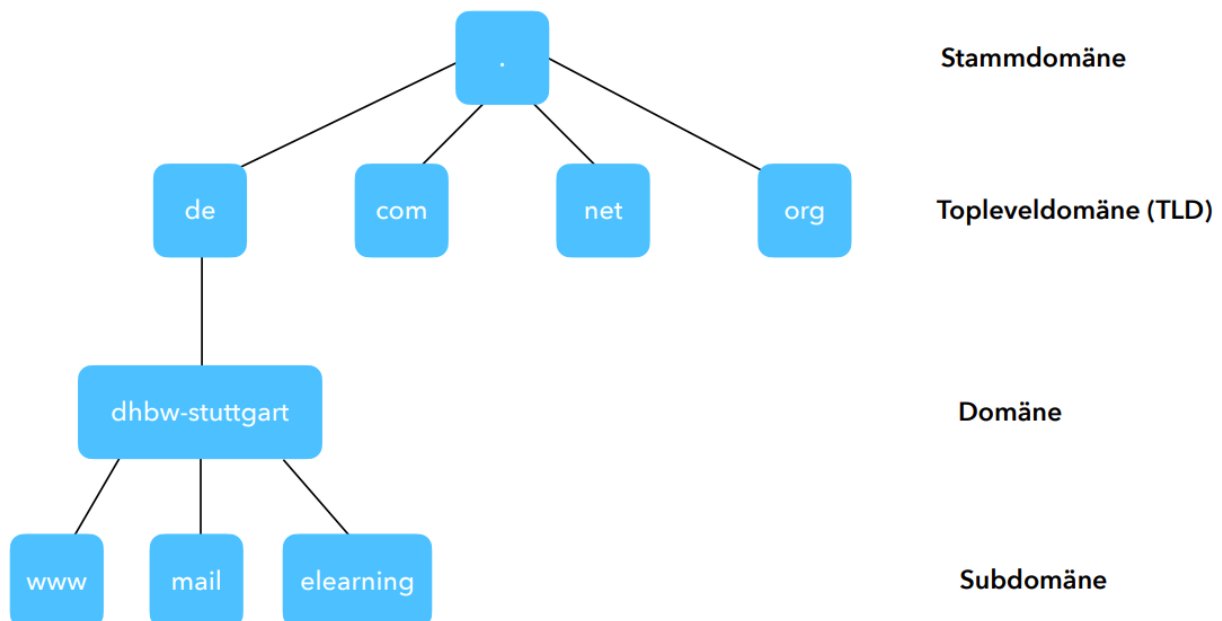
5.3 Transmission Control Protocol - TCP

Größte Schwachstelle von IP:

- Pakete können verloren gehen, dupliziert werden oder zu spät ankommen
- TCP ist eine Schicht auf der IP-Schicht, welche die Schwachstelle behebt (Kommunikation ist zuverlässig, es kann eine beliebige Menge von Daten transportiert werden und fehlerhafte Übertragungen werden wiederholt)
- Wenn keine Verbindung notwendig ist, kann auch UDP genutzt werden. Es definiert ein simpleres Protokoll welches Checksummen verwendet um Transportfehler zu melden.

6 Domain Name System - DNS

Beschreibt eine Tabelle welche eine Domain auf eine IP-Adresse auflösen kann. Domains werden hierarchisch aufgeteilt in:



6.1 Einordnung der Begriffe

DNS Server: Ein Computer der einen DNS Dienst ausführt und Client Anfragen beantwortet

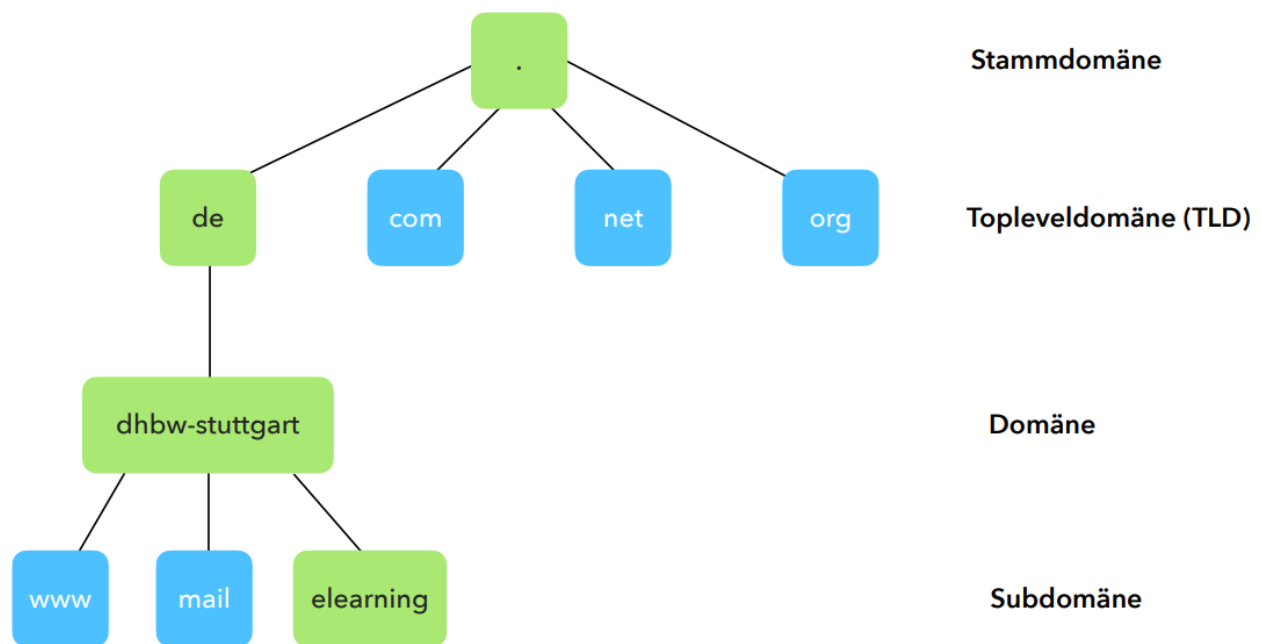
DNS Resolver: sozusagen die Clients, die DNS-Abfragen auf den Servern tätigen. Resolver können auf Clients oder auf Server ausgeführt werden

Resource records: Grundlegende Informationseinheit im DNS. Diese werden für die Verarbeitung der Clientanfragen genutzt. DNS-Server halten n RRs für den Anteil des Namespaces, für den sie autorisiert sind

Zones: Anteile des Namespaces für den der Nameserver autorisiert ist. Ein DNS-Server kann für mehrere Zonen autorisiert sein

6.2 DNS Zonen

Eine DNS Zone ist ein Teil des DNS-Namespaces, der von einer Organisation oder Person verwaltet wird. Zonen können von mehreren Servern verwaltet werden (primär oder sekundär).



Es gibt zwei Arten:

Forward Lookup Zonen

Informationen wie Namen werden in IP-Adressen aufgelöst. PTR Records können nicht auftreten (sind Reverse Lookup Zonen vorbehalten) und müssen mindestens einen NS und einen SOA Record enthalten.

Reverse Lookup Zonen

Enthalten Informationen wie IPs zu Namen aufgelöst werden. Stellen innerhalb des DNS-Namespace die Sonderdomäne `in-addr.arpa` dar. Können nur NS, SOA, PTR und CNAME Records enthalten.

Häufig zu einer Forward Lookup Zone auch eine Reverse Lookup Zone.

6.3 DNS Server

DNS Server bzw. Nameserver sind Server, die einer URL die richtige IP zuweisen können oder einer IP die richtige URL. Bedient Anfragen von Clients. Ein DNS Server kann primärer und sekundärer Server für andere Zonen zugleich sein. Verwaltet ausschließlich primäre Zonen. Sekundärserver für Ausfallsicherheit sinnvoll. Einem Namenseintrag können mehrere IP-Adressen zugeordnet werden. DNS-Server verteilen mit Round-Robin Verfahren.

Mehr zu DNS Servern findet ihr im Skript Seite 122 bis 137 - Finde ich nicht sinnvoll zu lernen. Müsst ihr wissen... Auf Nachfrage kann ich es noch rein packen

6.4 DNS Records

Ein DNS Record ist die kleinste Instanz, die die genaue Information enthält, welcher Domainname zu welcher Ziel-Adresse «gehört». So muss z.B. bei Hostpoint im DNS hinterlegt sein, dass «hostpoint.ch» auf die IP-Adresse «217.26.48.170» auflösen muss. Dies können Sie sich wie ein einfacher Texteintrag in einer Datei oder einer Datenbank vorstellen.

Der Record wird aber noch mit zusätzlichen Informationen versehen:

Hostname	TTL (Time to Live)	Klasse (hier Internet)	Typ	Ziel
hostpoint.ch.	300	IN	A	217.26.48.170

Natürlich gibt es noch weitere Typen von DNS Records, hier eine Auflistung der gängigsten:

Typ	Beschreibung	Inhalt
SOA	Start of Authority	ns.hostpoint.ch. hostmaster.hostpoint.ch. 20130614 86400 7200 3628800 12096000
NS	Autoritative Nameserver der Domain	ns.hostpoint.ch
A	IPv4 Adresse der Domain	217.26.48.170
AAAA	IPv6 Adresse der Domain	2a00:d70:0:a::333
CNAME	Kanonischer Name einer Domain	Host
MX	Mail-Exchange, der für Ihre Domain zuständige Mailserver	mx1.mail.hostpoint.ch mx2.mail.hostpoint.ch

7 Webserver

Verarbeitet die Anfrage des Clients und gibt entsprechende Resource zurück. Ist ein Serverdienst, der Informationen über das HTTP Protokoll zur Verfügung stellt. Bekannteste Webserver sind NGINX und Apache.

7.1 SSL/TLS Verschlüsselung

TLS ist das Verbesserte SSL. Ein Netzwerkprotokoll zum Verschlüsseln und Authentifizieren von Daten zwischen Client und Webserver. Verwendet Public-Key-Infrastruktur. Erweiterung des TCP/IP um fiktive Schichten, die sich zwischen Transport - und Anwendungsschicht schieben. Basiert auf dem TLS Handshake Layer.

7.1.1 Symmetrische Verschlüsselung

Schlüssel für Ver- und Entschlüsselung sind identisch.

7.1.2 Asymmetrische Verschlüsselung

Schlüssel für Ver- und Entschlüsselung sind unterschiedlich. Mit öffentlichem Schlüssel wird verschlüsselt. Mit privatem Schlüssel entschlüsselt. Man versendet nur seinen Schlüssel zum Verschlüsseln. Den zum Entschlüsseln behält man.

7.1.3 Public-Key Infrastruktur

Public-Key Infrastruktur (PKI) bezeichnet ein System des Ausstellens, Verteilen und Überprüfen von digitalen Zertifikaten. Certificate Authorities (CAs) erstellen Zertifikate für Dritte und prüfen und validieren die Echtheit der natürlichen/juristischen Person.

Ein SSL Zertifikat wird also an eine juristische/natürliche Person gebunden. Damit ist sichergestellt, dass nur die entsprechende Person Daten entschlüsseln kann, die mit dem öffentlichen Schlüssel verschlüsselt wurden. Dient zur Authentifizierung und Verschlüsselung. Ein solches Zertifikat muss beantragt werden und wird dann durch CA geprüft. Zertifikat wird nun auf dem Server installiert. Beim surfen über eine Webseite wird das hinterlegte Zertifikat heruntergeladen. Die Signatur kann mit dem öffentlichen Schlüssel der CA geprüft werden.

7.1.4 TLS Handshake

TLS ist ein Verschlüsselungs- und Authentifizierungsprotokoll, das zur Sicherung der Internetkommunikation entwickelt wurde. Ein TLS-Handshake ist der Prozess, der eine Kommunikationssitzung mit TLS-Verschlüsselung startet. Während eines TLS-Handshakes tauschen die beiden kommunizierenden Seiten Nachrichten aus, um sich gegenseitig zu bestätigen, sich gegenseitig zu verifizieren, die von ihnen verwendeten kryptographische Algorithmen festzulegen und Sitzungsschlüssel zu vereinbaren. TLS-Handshakes sind ein grundlegender Bestandteil der Funktionsweise von HTTPS.

Ein TLS-Handshake erfolgt immer dann, wenn ein Nutzer per HTTPS eine Website ansteuert und der Browser beginnt, Abfragen an den Ursprungsserver zu senden. Zu einem TLS-Handshake kommt es auch bei sonstigem Informationsaustausch per HTTPS, etwa bei API-Aufrufen und DNS-Abfragen über HTTPS.

Ablauf eines TLS Handshakes:

1. **Client Hello:** Der Client initiiert den Handshake indem er eine (Hello) Nachricht an den Server sendet. Diese Nachricht enthält, welche TLS Version und Cipher Suites er unterstützt. Außerdem einen Zufallsstring (Client Random).
2. **Server Hello:** Der Server antwortet auf das Client Hello mit dem von ihm gewählten Cipher und TLS Version sowie einen Zufallsstring (RNs)
3. **Server Certificate:** : Unmittelbar nach dem Server Hello sendet der Server das Server Zertifikat (inkl. des Public Key des Zertifikats.)
4. **Client prüft Zertifikat:** Das versichert dem Client, dass er mit dem tatsächlichen Eigentümer der Domain interagiert.
5. **Pre-master-secret:** Der Client generiert eine Zufallsnummer, das so genannte Pre-master Secret. Diese wird mit dem public key des Servers verschlüsselt und an diesen gesendet. Damit ist sichergestellt, dass nur der Server dieses lesen kann.
6. **Authentifizierung:** Der Server entschlüsselt mit seinem private key den vom Client gesendeten premaster-secret. Der Client ist damit authentifiziert.
7. **Sitzungsschlüssel:** Client als auch Server erzeugen aus RNC, RNs und PMS den Sitzungsschlüssel. Beide sollten hier zum selben Ergebnis kommen.
8. **Fertig(Client):** Der Client sendet an den Server eine Fertig-Nachricht, die mit dem MS verschlüsselt ist
9. **Fertig(Server):** Der Server sendet an den Server eine Fertig-Nachricht, die mit dem MS verschlüsselt ist. Der Handshake ist damit abgeschlossen und die Kommunikation wird mit den Sitzungsschlüsseln (master-secret) fortgesetzt.

8 HTTP

Das Hypertext Transfer Protocol (HTTP) ist ein einfaches request/response Protokoll. Client sendet Request und Server antwortet mit Response. Basierend auf TCP ist HTTP zuverlässig. HTTP ist zustandslos. Nach Beendigung der Anfrage besteht keine Verbindung zwischen Client und Server.

Die Requests können durch verschiedene Methoden gekennzeichnet werden. Meistens sind diese jedoch POST, PUT, CONNECT oder GET. Bei GET werden die zu übergebenen Parameter im URL übertragen. Bei POST werden die zu übergebenen Daten im Body mitgegeben.

Die Response kann durch Statuscodes anzeigen wie der Status der Request aussieht.

8.1 Die Rollen in HTTP

Client, User Agent: Programm das Anfragen absendet.

Server: Programm welches den Request annimmt und antwortet.

Proxy: Ein Programm welches „im Auftrag“ des Servers agiert.

- z.B. wenn angefragt Ressource auf Proxy verfügbar ist
- Client sendet Anfrage an Proxy, nicht an Server.
- Proxy leitet Anfrage an Server weiter, wenn dieser nicht selbst verarbeitet werden kann.

Gateway: Eine Software die als Vermittler für anderen Server agiert.

- Im Gegensatz zum Proxy agiert der Client mit dem Gateway als wenn es der Server wäre.
- Kann auch als Dolmetscher fungieren.

Tunnel: Programm das „blind“ zwischen den beiden Kommunikationsteilnehmer agiert.

- z.B. Firewalls

8.2 Persistente Verbindungen

Es können persistente Verbindungen aufgebaut werden. Diese können jederzeit von beiden Seiten unterbrochen werden.

8.3 Chunked Transfer

Oftmals ist die Größe der zu übertragenen Entität nicht im Voraus bekannt. Bspw. Durch in CGI oder Servlet das die Entität dynamisch generiert. Jedes Stück (Chunk) wird separat transferiert. Ende wird durch einen „null großen“ Transfer beendet.

8.4 URL

SCHEMA://[USER[:PASSWORT]@]HOST[:PORT]/PFAD?PARAMETER

Zum Beispiel: <http://admin:myPassword@dhw-stuttgart.de:443/seite2?login=ja>

<http://dhw-stuttgart.de>

8.5 HTTP Caching

Cache wird dazu verwendet um die Responses eines Server zwischenspeichern. Ziel: Antwortzeiten verbessern. Gecacht werden kann clientseitig, serverseitig oder separat durch zwischenliegende Komponenten. Um gecachte Inhalte aktuell zu halten wird der Cache Inhalt mit einer life-time versehen. Diese zeigt an wie lange der Cache bestehen bleiben soll.

8.6 HTTP Cookies

HTTP ist zustandslos! Jeder Request ist unabhängig von den folgenden Requests. Ein Cookie ist eine Datei die auf dem Rechner des Clients gespeichert wird und eine Referenz zu

einer URL (des Servers) hält.

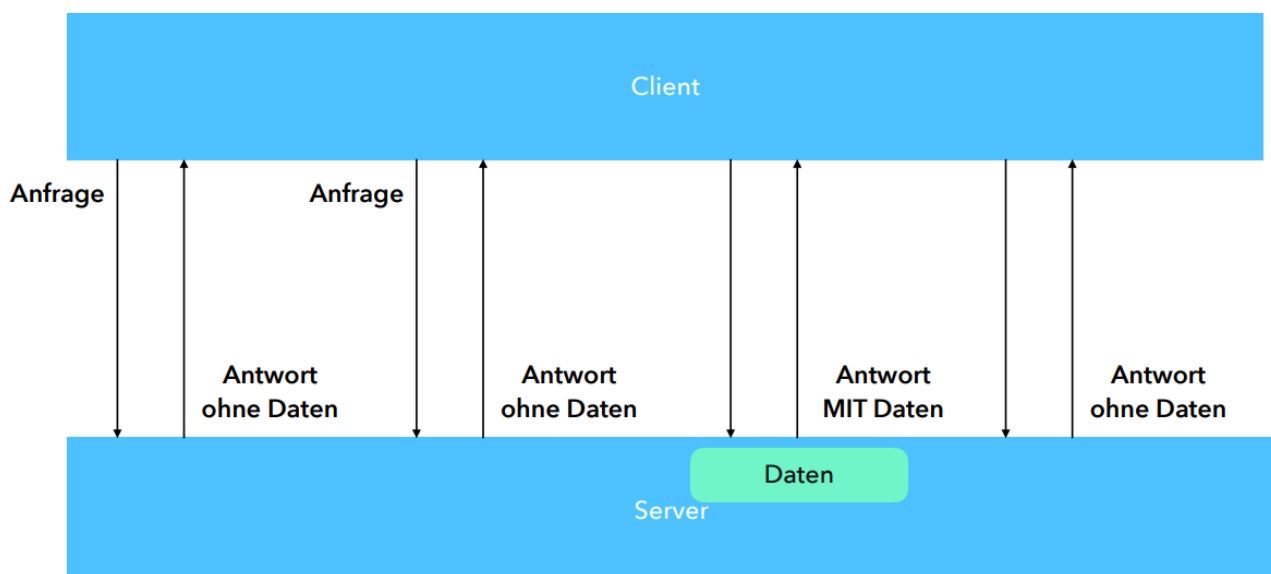
9 Kommunikation

HTTP Kommunikation ist unidirektional. Moderne Anwendungen benötigen auch Kommunikation vom Server zum Client.

9.1 Polling

Client fragt im Hintergrund beim Server in regelmäßigen Zeitabständen nach neuen Daten an.

- Vorteile:
 - Benutzer muss die Webseite nicht neu laden um neue Informationen zu bekommen.
 - Gefühlte Live-Interaktion
- Nachteile:
 - Relativ viele HTTP-Anfragen müssen an den Server gesendet werden, auch wenn dort ggf. keine Informationen vorliegen.



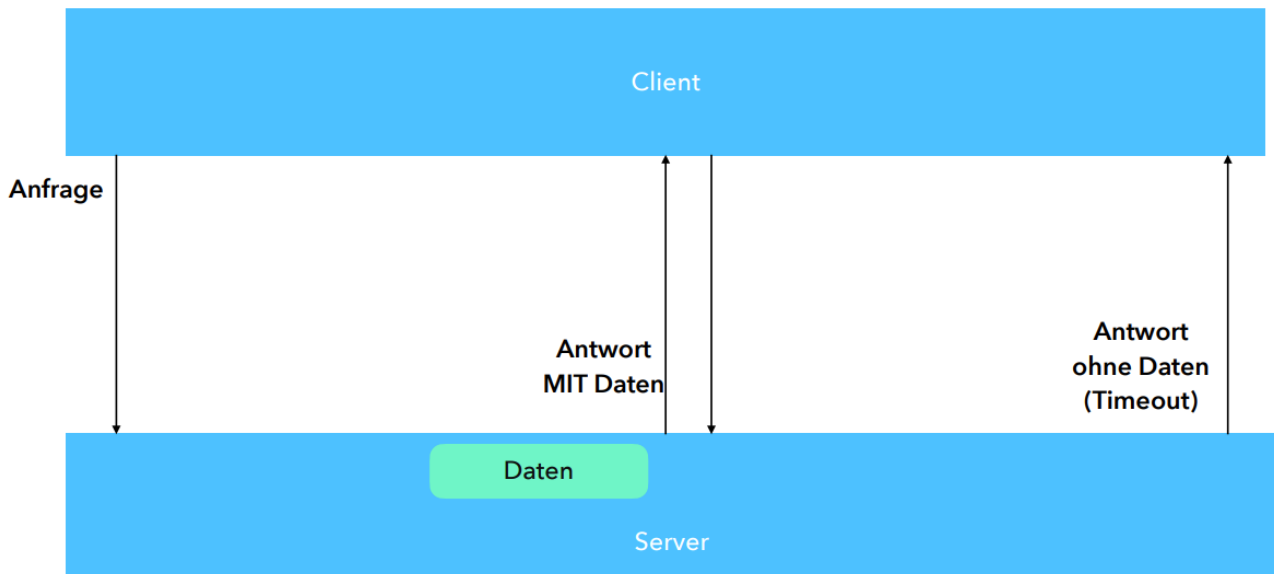
Polling wird clientseitig durch JavaScript realisiert.

9.2 Long Polling

Long Polling versucht den Nachteil des Pollings auszugleichen. Client sendet Anfrage und hält die Anfrage so lange offen bis ein Timeout geschieht oder neue Daten vorliegen.

- Vorteile:
 - Weniger Requests, daher weniger Netzwerklast

- Nachteile:
 - Ggf. ressourcenintensiv, da je nach Besucherzahl, für jeden Besucher die Kommunikation bestehen bleiben muss.

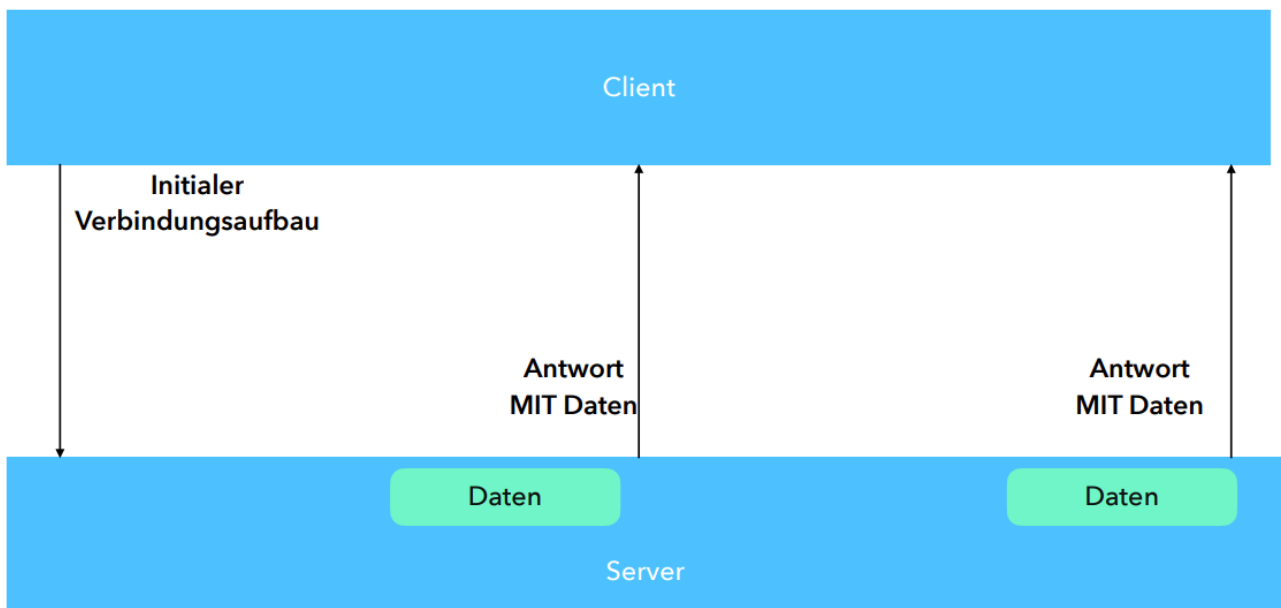


Long Polling wird clientseitig durch JavaScript welches im Browser läuft und in einem definierten Zeitintervall Abfragen beim Server durchführt implementiert. Serverseitig: Verbindung wird im Backend bspw. durch eine n Sekunden lange Schleife offengehalten.

9.3 Server-Sent Events - SSE

Technologie bei der der Server über eine HTTP Verbindung aktiv Daten an den Client senden kann. Client muss nächst mit Hilfe von JavaScript eine Verbindung zum Server herstellen. Nach Verbindungsaufbau kann Server zum Client Informationen senden. Verbindung unidirektional - allerdings vom Server zum Client.

- Vorteile:
 - Eigenen sich gut für einseitige Kommunikation
 - Keine Absicherung notwendig gegen fehlerhafte Daten vom Client
 - Keine unnötige Kommunikation
- Nachteile:
 - Einseitige Kommunikation

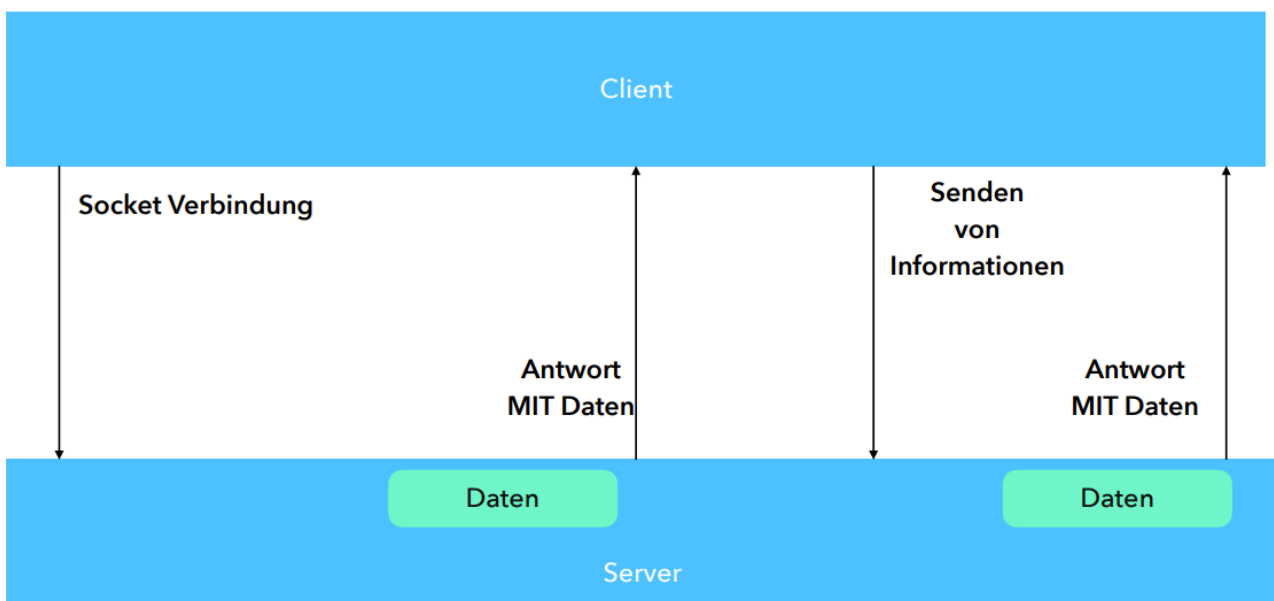


Die Realisierung erfolgt durch Frontend Integration ähnlich wie bei Websockets. Es existieren Bibliotheken im Frontend. Backend antwortet mit Content-type: text/event-stream.

9.4 WebSockets

Auf Basis eines TCP basierenden Netzwerkprotokolls. WebSocket-Server und WebSocket-Client ist notwendig. Client eröffnet WebSocket-Verbindung zum Server. Bidirektionale Verbindung.

- Vorteile:
 - Schnelle live Interaktion zwischen Client und Server bidirektional
- Nachteile:
 - Absicherung der Verbindung muss ggf. zusätzlich geschehen
 - Ältere Browser unterstützen WebSocket nicht



Die Realisierung erfolgt durch WebSocket Server im Backend und JavaScript im Frontend.

10 Webformate

10.1 CSV

Einfach strukturierte Datensätze, Tabellarischer Inhalt, Datensätze (Zeilen) standardmäßig über Zeilenumbruch eingeleitet, Komma oder Semikolon standardmäßig als Trennsymbol einzelner Spalten, Optional erste Zeile als Spaltennamen

Beispiel:

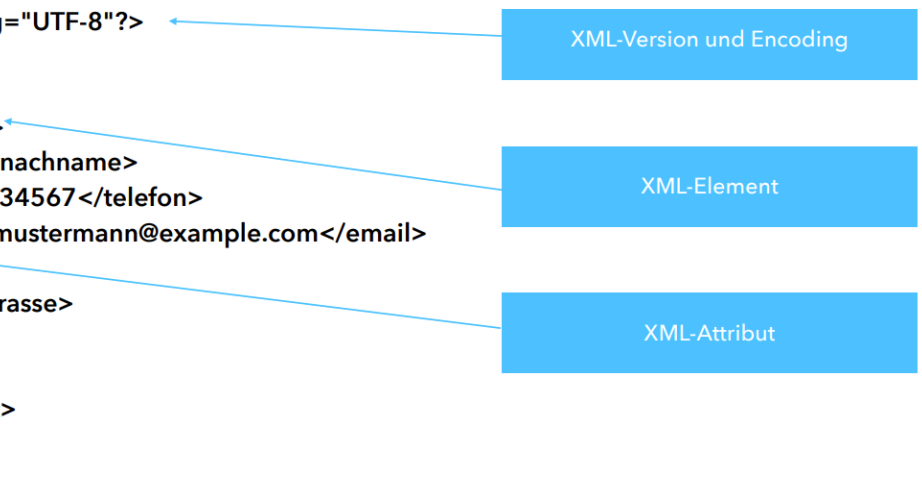
Vorname,Nachname,Strasse,PLZ,Ort Max,Mustermann,Musterstraße 44,70736,Fellbach Anna,Musterfrau,Musterstraße 33,71332,Waiblingen

10.2 XML (Extensible Markup Language)

Auszeichnungssprache (Markup Language), Hierarchische Strukturierung möglich, XML-Elemente und XML-Attribute vorhanden, XML-Attribute spezifizieren XML-Elemente genauer, XML beliebig erweiterbar (Extensible).

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<kontakte>
  <kontakt>
    <vorname>Max</vorname>
    <nachname>Mustermann</nachname>
    <telefon type="mobil">01234567</telefon>
    <email type="privat">max.mustermann@example.com</email>
    <adresse>
      <strasse>Musterstraße</strasse>
      <nummer>99</nummer>
      <plz>12345</plz>
      <stadt>Musterstadt</stadt>
    </adresse>
  </kontakt>
```



Zur Verarbeitung von XML gibt es sogenannte XML-Parser.

10.2.1 XML Schema

XML-Schema definieren wie ein XML Dokument aufgebaut sein darf.

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="kontakte">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="kontakt" maxOccurs="unbounded" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:element type="xs:string" name="vorname"/>
              <xs:element type="xs:string" name="nachname"/>
              <xs:element type="xs:string" name="telefon"/>
              <xs:element type="xs:string" name="email"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

10.3 JSON

Einfacher Aufbau und leichte Integration in JavaScript-Anwendungen, Strukturierte Definition von Daten, Schlanker als XML, Objekteigenschaften (Schlüssel) werden in doppelten Anführungszeichen schreiben und durch einen Doppelpunkt von dem jeweiligen Wert getrennt, Arrays werden in eckigen Klammern notiert []

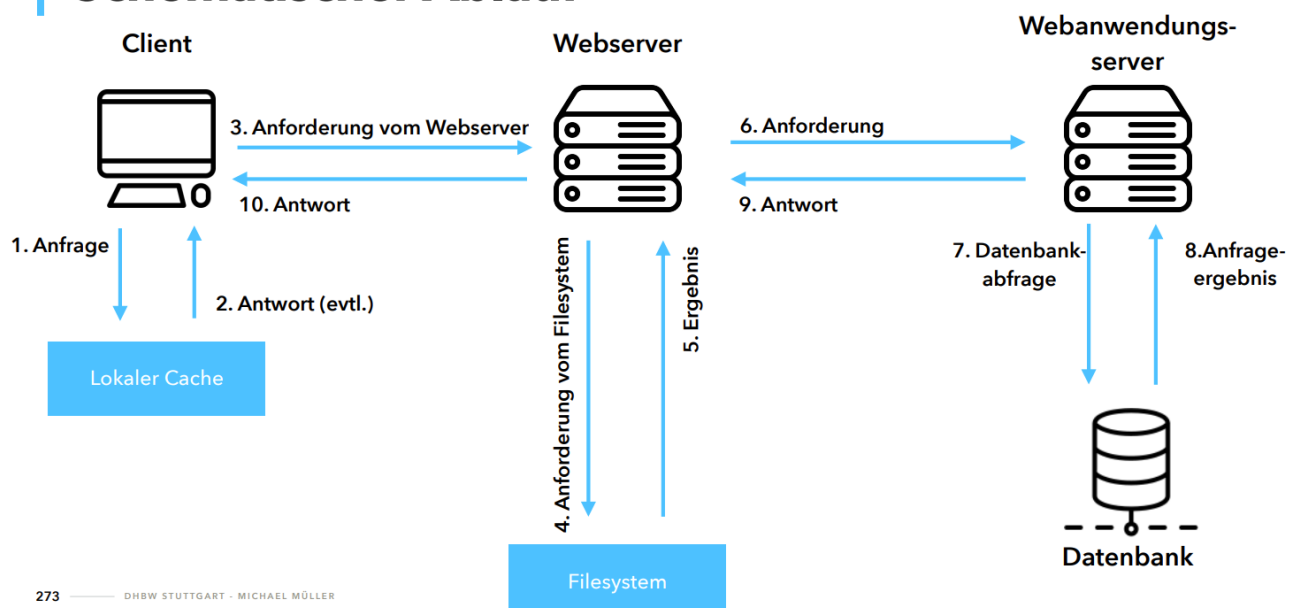
```
{
  "kontakte" : [
    {
      "vorname" : "Max",
      "nachname" : "Mustermann",
      "telefon" : {
        "type" : "mobile",
        "number" : "2932323"
      },
      "email" : "max@mustermann.de",
      "address" : {
        "street" : "Musterstraße",
        "number" : 99,
        "plz" : 72323,
        "ort" : "Musterstadt"
      }
    },
    {
      "vorname" : "Petra",
      "nachname" : "Musterfrau",
      "telefon" : {
        "type" : "mobile",
        "number" : "2242443"
      },
      "email" : "petra@mustermann.de",
      "address" : {
        "street" : "Musterstraße",
        "number" : 23,
        "plz" : 212414,
        "ort" : "Musterstadt"
      }
    }
  ]
}
```

Im Falle von JavaScript ist bspw. das Parsen von JSON-Dokumenten sogar nativ in die Sprache eingebaut. Analog zum XML-Schema gibt es auch für das JSON-Format die Möglichkeit ein Schema zu definieren. Wie auch bei dem XML gibt es auch bei JSON entsprechende Validatoren.

11 Serverseitige Programmierung

...bedeutet, dass Programme serverseitig ausgeführt werden, die einen dynamischen Inhalt generieren, bevor diese zum Client gesendet wird.

Schematischer Ablauf



Kann durch sämtliche Programmiersprachen umgesetzt werden. Kompilierte Programmier-

sprachen(C#, Java, ...) müssen vorher kompiliert werden und interpretierte werden zur Laufzeit interpretiert(Python, PHP, ...).

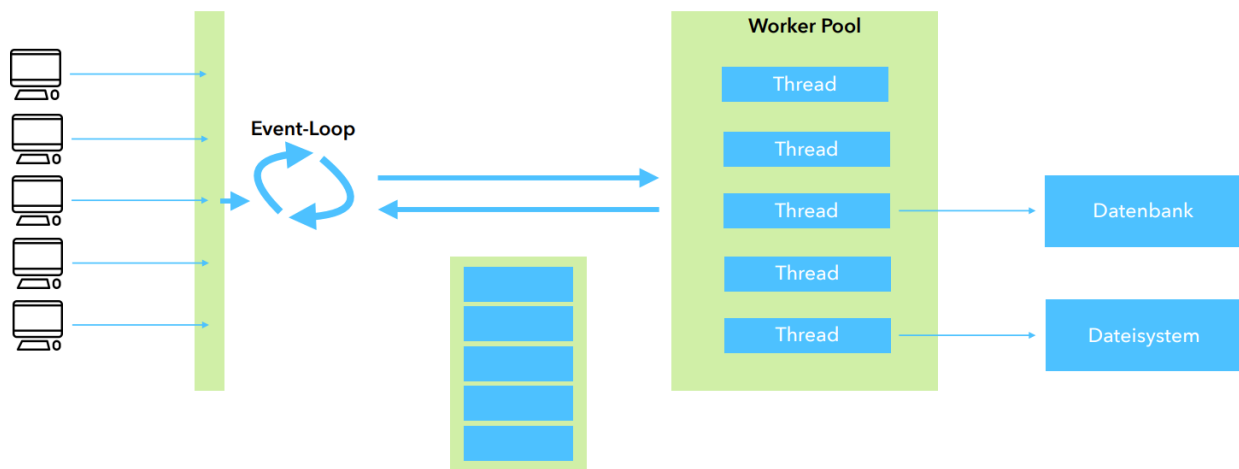
Imperative Programmierung definiert wie ein Programm funktioniert(genaue Abläufe), während deklarative Programmierung definiert was ein Programm macht. Dem Computer wird vermittelt, was man erreichen möchte.

12 Node.JS

Laufzeitumgebung auf Serverseite zum Betrieb von Netzwerkanwendungen, Basiert auf der Google V8-Javascript Engine, Packetmanager NPM, Node.JS enthält im Standard bereits viele Funktionen und APIs.

Vorteil: Ist sowohl für das Frontend als auch das Backend geeignet ist.

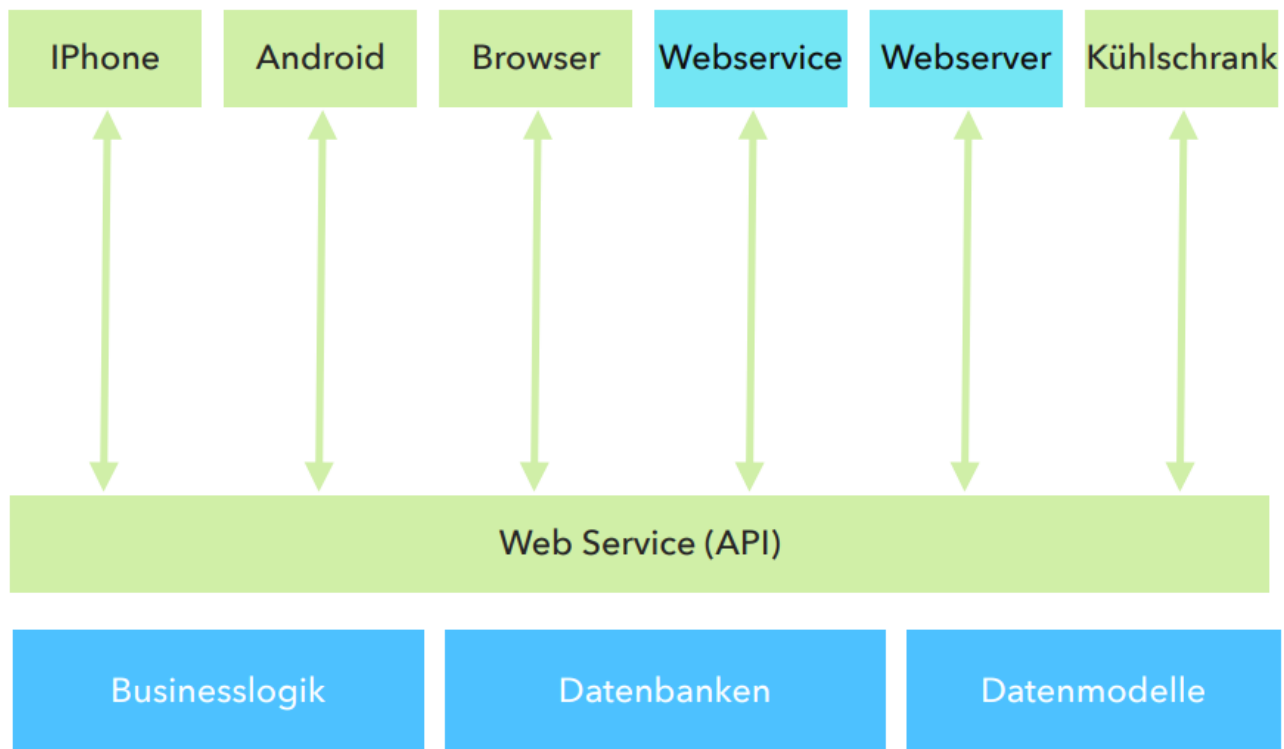
Node.JS Architektur



13 Web Services

Ist eine im Internet veröffentlichte Software die über Schnittstellen angesprochen werden kann. Interaktion zwischen Client und Server geschieht durch den Austausch von (ausgehandelten) Nachrichten über Internetprotokolle. Die Technologien hinter den Web Services sind plattformunabhängig. Mit Web Services können Dienste bereitgestellt werden, die Webseiten, mobile Apps oder Anwendungen nutzen, z.B. auch ein Kühlschrank.

Das heißt z.B. können Anwendungen in Python mit Anwendungen in node.JS oder Java miteinander kommunizieren und das obwohl diese auf unterschiedlichen Servern laufen.



SOAP

- Simple Object Access Protocol
- Kommunikation findet durch den Austausch von XML Nachrichten statt
- SOAP definiert die Kommunikationsregeln, sowie Aufbau des XML Dokuments.

REST

- Representational State Transfer
- ist eine Architektur
- Wird auf die Methoden GET, POST, PUT, DELETE beschränkt

GraphQL

- Abfragesprache für APIs
- serverseitige Laufzeitumgebung zum Ausführen von Abfragen

13.1 SOAP

SOAP ist statuslos und ein Standardprotokoll, das zunächst entwickelt wurde, damit Anwendungen, die mit verschiedenen Sprachen und auf verschiedenen Plattformen erstellt wurden, miteinander kommunizieren konnten. Da es sich um ein Protokoll handelt, umfasst es integrierte Regeln, die Komplexität und Overhead erhöhen, was zu längeren Seitenladezeiten führen kann. Diese Standards bieten jedoch integrierte Compliance, die es für Unternehmensszenarien attraktiv macht. Zu den integrierten Compliance-Standards gehören Sicherheit, Atomizität, Konsistenz, Isolation und Dauerhaftigkeit (ACID), eine Reihe von Eigenschaften zur Gewährleistung zuverlässiger Datenbanktransaktionen.

Zu den üblichen Webservice-Spezifikationen gehören:

WS-Security (Web Services Security): Standardisiert, wie Nachrichten durch eindeutige Kennungen, sogenannte Token, gesichert und übertragen werden.

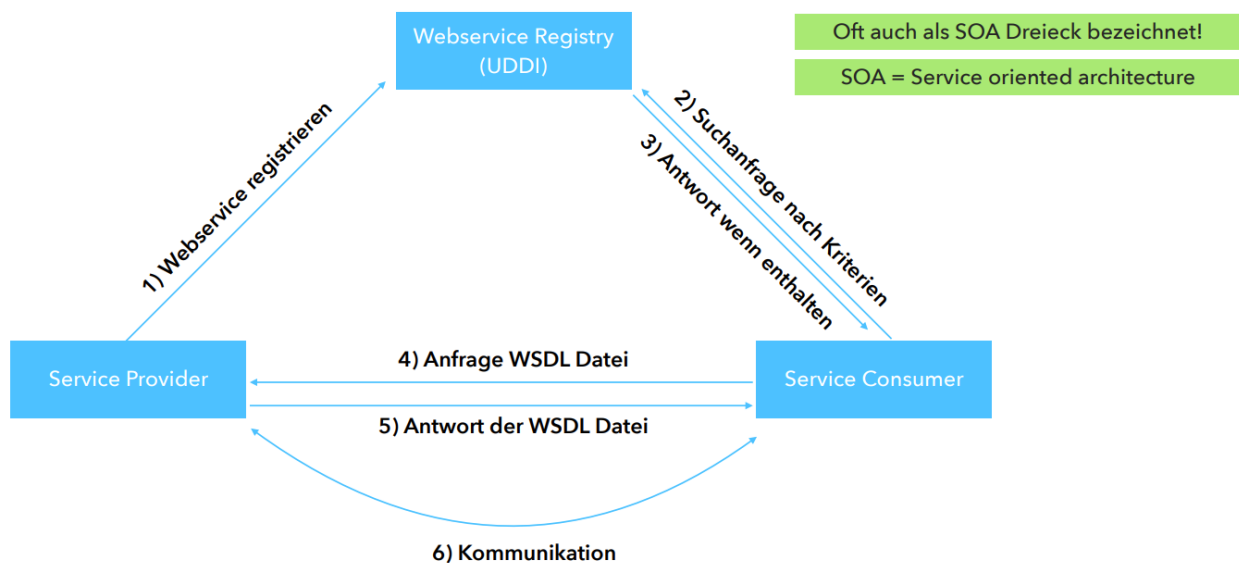
WS-ReliableMessaging: Standardisiert die Fehlerbehandlung zwischen Nachrichten, die über eine unzuverlässige IT-Infrastruktur übertragen werden.

WS-Addressing (Web Services Addressing): Paketierte Routing-Informationen als Metadaten in SOAP-Headern, statt diese Informationen tiefer im Netzwerk zu verwalten.

WSDL (Web Services Description Language): Beschreibt, was ein Webservice tut und wo dieser Service beginnt und endet.

Wenn eine Datenanforderung an eine SOAP-API gesendet wird, kann sie über eines der Protokolle der Anwendungsschicht verarbeitet werden: HTTP (für Webbrowser), SMTP (für E-Mail), TCP und andere. Sobald eine Anforderung jedoch empfangen wird, müssen SOAP-Nachrichten als XML-Dokumente zurückgegeben werden. Eine abgeschlossene Anfrage an eine SOAP-API kann nicht in einem Browser zwischengespeichert werden. Daher kann später auch nicht mehr auf die Anfrage zugegriffen werden, ohne sie erneut an die API zu senden.

SOAP - Workflow



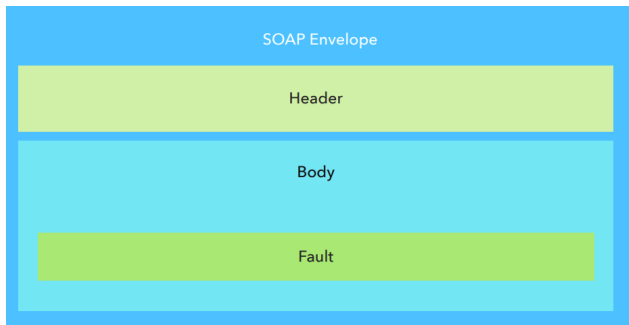
13.1.1 SOAP Nachrichten

Struktur:

- SOAP Envelope
 - <Envelope> ist das Wurzelement in jeder SOAP Nachricht. Es enthält den optionalen Header und den Body
- SOAP Header
 - Anwendungsbezogene Informationen, z.B. Authentifizierung
 - Definition des nächsten Hops (Intermediary)
 - Definition des Ziels (Ultimate Reciever)
- SOAP Body
 - Enthält die Daten für Empfänger
 - Informationen und durch den Ultimate Reciever zu interpretieren und zu verarbeiten

- SOAP Fault

- Ist ein Unterelement des Body und kann zum Melden von Fehlern genutzt werden.



```
<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:ns="http://www.irgendwas.de">

  <soap:Header>
    <ns:dateAndTime>2021-03-01T15:40:00</ns:dateAndTime>
  </soap:Header>
  <soap:Body>
    <ns:getUhrzeit>
      <ns:land>Germany</ns:land>
    </ns:getUhrzeit>
  </soap:Body>

</soap:Envelope>
```

13.1.2 SOAP - WSDL

WSDL steht für Web Services Description Language. XML-basierte Sprache, mit der man die von einem Webservice angebotenen Operationen beschreiben kann, also welche Parameter und Rückgabewerte erwartet werden können.

SOAP - WSDL Aufbau

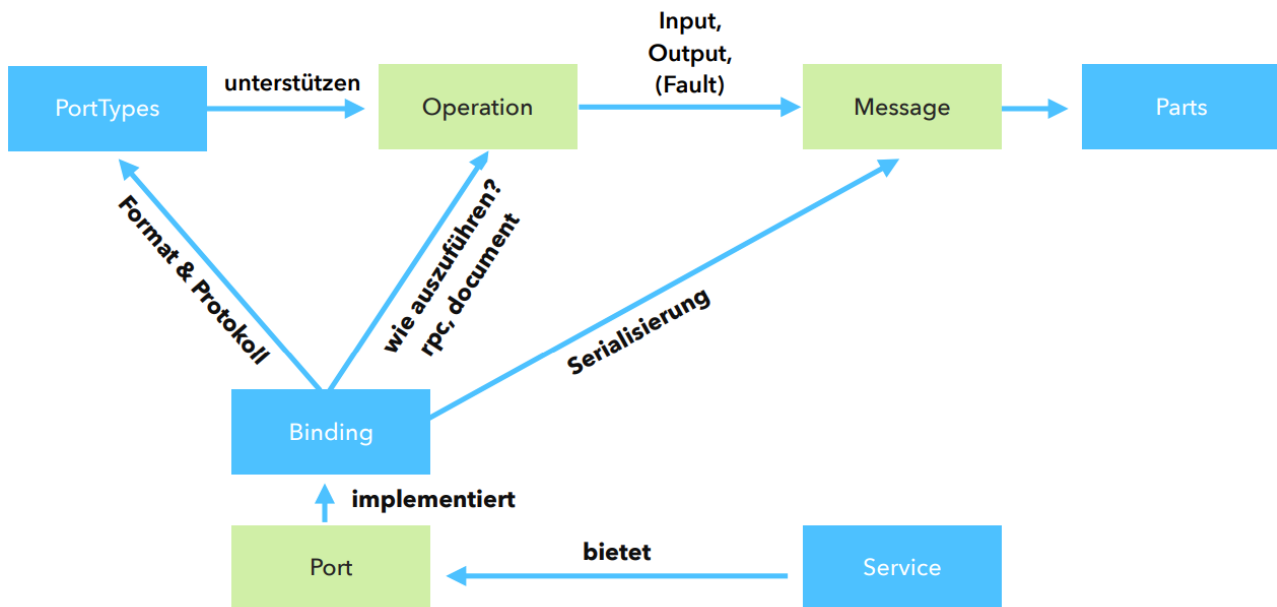
```
<types>
  <schema targetNamespace="http://irgenwas.de/rechner.xsd"
    xmlns="http://www.w3.org/2000/10/XMLSchema">
    <element name="ZeitAnfrage">
      <complexType>
        <choice>
          <element name="zeitzone" type="string"/>
          <element name="land" type="string"/>
        </choice>
      </complexType>
    </element>
    <element name="Uhrzeit">
      <complexType>
        <all>
          <element name="zeit" type="date"/>
        </all>
      </complexType>
    </element>
  </schema>
</types>
<!-- MESSAGE -->
<!-- PORTTYPE -->
.....
```

```
<?xml version="1.0" ?>
<definitions name="NAME" targetNamespace="URI">

  <import namespace="uri" location="uri" />*
  <documentation />?
  <types?>
  <message name="NAME">*>
  <portType name="NAME">*>
  <binding name="NAME" type="QNAME">*>
  <service name="NAME">*>

</definitions>
```

- Types werden genutzt um Nachrichtenformate zu definieren.
- Diese Nachrichtenformate werden dann von den "Messages" verwendet.



13.1.3 UDDI

Universal Description, Discovery and Integration = UDDI ist eine Möglichkeit, Informationen über Web-Services zu publizieren und zu ermitteln. UDDI hat zwei Funktionen: Es handelt sich um ein SOAP-basiertes Protokoll, das definiert, wie Clients mit UDDI-Registries kommunizieren und es handelt sich um eine bestimmte Gruppe global replizierter Registries. Unterschieden wird in:

- White Pages:
 - Geben Auskunft über die Identität des Serviceanbieters. Geschäfts- und Kontaktdaten und eine eindeutige Unternehmenskennzahl (DUNS)
- Yellow Pages:
 - Die Webservices von dem in den White Pages gelisteten Anbieters.
 - Diese werden nach diversen Kriterien hinsichtlich Anwendungszwecks kategorisiert.
- Green Pages:
 - Konkrete Schnittstellen Beschreibungen der Webservices. Das sogenannte tModel.

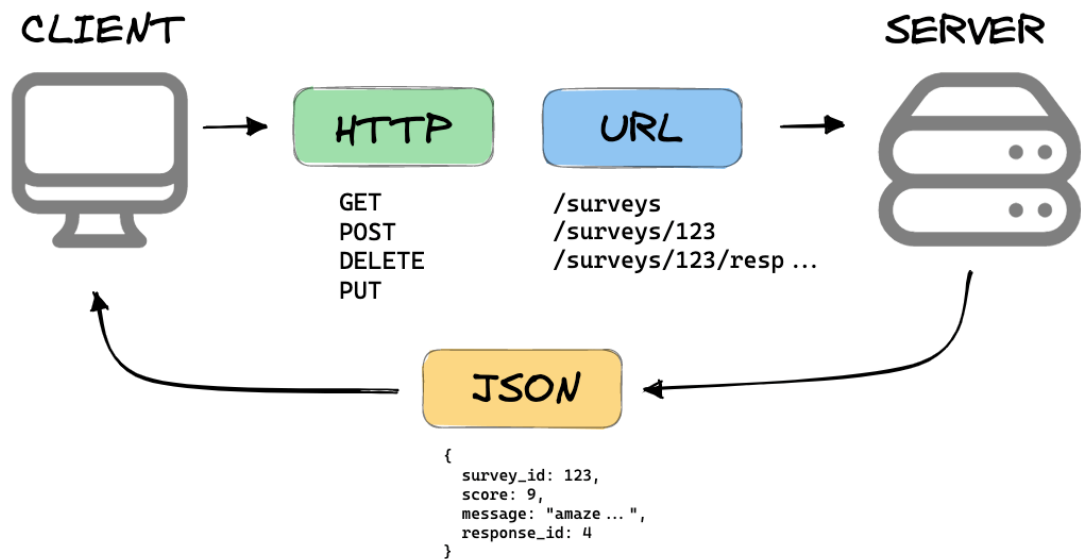
13.2 REST

REpresentational State Transfer (REST). REST ist eine Architektur, keine Technologie, und ist ein Paradigma für die Softwarearchitektur von verteilten Systemen, insbesondere für Webservices. REST ist eine Abstraktion der Struktur und des Verhaltens des World Wide Web. REST hat das Ziel, einen Architekturstil zu schaffen, der den Anforderungen des modernen Web besser genügt.

REST schreibt vor, dass jede Ressource eindeutig durch eine URI identifiziert und gefunden werden kann. Ebenso bedient sich REST nur an den CRUD Methoden:

PUT - **C**REATE**GET** - **R**EAD**POST** - **U**PGRADE**DELETE** - **D**ELETE

WHAT IS A REST API?



mannhowie.com

REST erreicht somit folgende Ziele:

- Geschwindigkeit
 - Caching durch eindeutige URLs
- Skalierbarkeit
 - Status ist in Request und Response enthalten. Damit kann durch mehrere Cache eine Skalierbarkeit erreicht werden.
- Einfachheit
 - Vier generische Interaktionen (Create, Retrieve, Update, Delete)
- Datenunabhängigkeit
 - Daten einer Resource können unterschiedlich repräsentiert werden. (Egal ob HTML, XML oder TEXT gesendet wird.)

13.3 GraphQL

GraphQL ist eine Abfragesprache. Response basiert auf JSON. Request ist ein eigenständiges Format (auch wenn es ähnlich wie JSON aussieht). Struktur die vom Server zurückgegeben werden, kann im Client formuliert werden. Client bestimmt also Aufbau der Antwort.

GraphQL - Query/Response

Client Request

```
{
  autoren {
    id,
    vorname,
    nachname,
    buch {
      titel,
      beschreibung
    }
  }
}
```

Server Response

```
{
  "data": {
    "autoren": [
      {
        "vorname": "Michael",
        "nachname": "Mueller",
        "buch": [
          {
            "titel": "Webengineering",
            "beschreibung": "Das ist ein Buch ..."
          }
        ]
      },
      {
        "vorname": "Max",
        "nachname": "Mustermann",
        "buch": [
          ...
        ]
      }
    ]
  }
}
```

385 — DHBW STUTTGART - MICHAEL MÜLLER

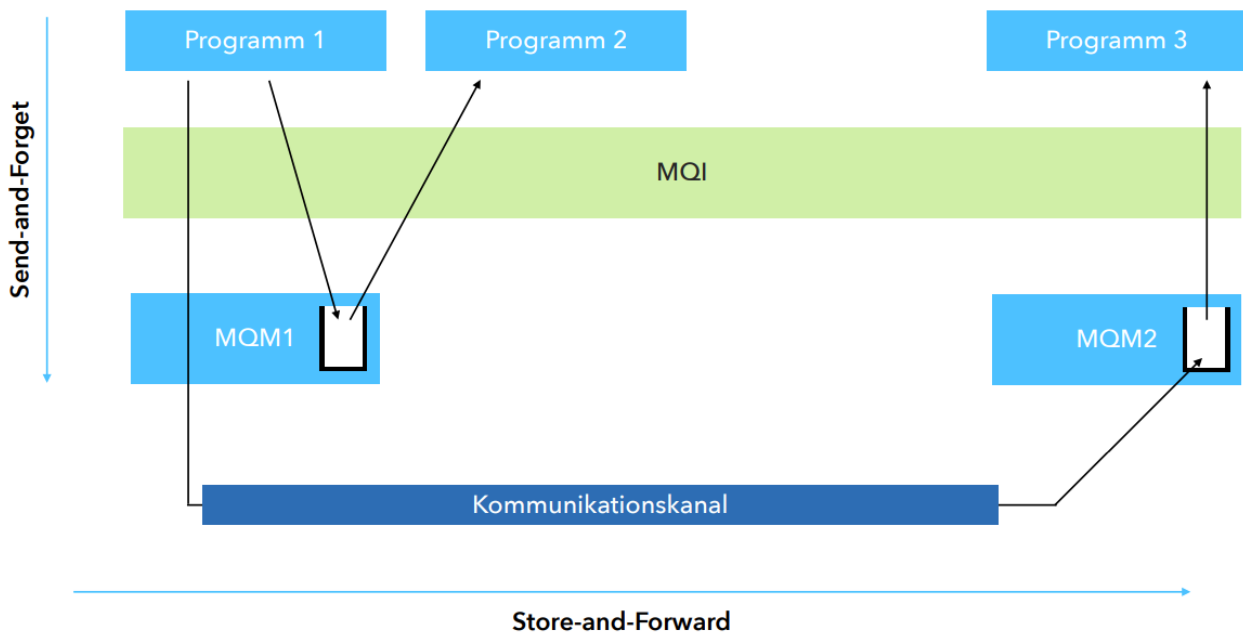
- Vorteile
 - Durch die verschachtelte Struktur können Abfragen erstellt werden, für die man mit REST mehrere Abfragen benötigen würde.
- Nachteile
 - Noch kein hoher Verbreitungsgrad, daher wenig Bibliotheken
 - Höherer Implementierungsaufwand als bei REST

14 Message Queueing

Messaging ist das Senden von Nachrichten zwischen zwei Komponenten. Messaging wird verwendet um Komponenten zu integrieren (In der Regel größere Einheiten). Aber: Datenaustausch, Remote Procedure Calls, geteilte Datenbanken, etc. können das auch. Message Queueing ermöglicht eine Kommunikation zwischen Computern unterschiedlicher Betriebssysteme oder unterschiedlicher Sprachen. Messaging Systeme können ein universaler Übersetzer zwischen Anwendungen sein, die eigenständig entstanden sind. Die Kommunikation erfolgt dabei über ein einheitliches, gemeinsames Muster. Message Queueing ist asynchron. Messaging erlaubt einen send & forget Ansatz zur Kommunikation. Der Sender muss nicht auf den Empfänger warten, bis dieser die Nachricht verarbeitet hat. Er muss auch nicht darauf warten, dass das Messaging System (MS) die Nachricht versandt hat. Der Sender muss ausschließlich darauf warten, dass das MS die Nachricht akzeptiert hat. Er kann weiterarbeiten, während die Nachricht im Hintergrund transferiert wird. Der Empfänger kann eine Empfangsbestätigung oder Ergebnis an den Sender zurücksenden. Dies benötigt eine neue Nachricht. (Der Sender muss hierzu ein Callback Mechanismus implementieren)

14.1 Message Queue Manager - MQM

Speichert die Queues(Warteschlangen), managed die Zugriffe, implementiert Sicherheit und Autorisierung und stellt ein Interface zur Verfügung. Anwendung die Message Queueing implementieren, müssen sich mit einem MQM verbinden.



15 Datenbanken

15.1 Relationale Datenbanken

Klassisch werden die Codd'schen Regeln verwendet und ACID Garantien für Transaktionen. SQL als Abfragesprache.

15.2 Nicht-Relationale Datenbanken - NoSQL

Nicht-Relationale Datenbanken setzen auf strukturierte Datenspeicher statt Tabellen. Sie organisieren bspw. Dokumente, Graphen, Spalten und Wertepaare. Sie lassen sich horizontal skalieren, also die Verteilung auf mehrere Server. Nicht-Relationale Datenbanken verwenden für die Verwaltung keine Relationen sondern andere Datenstrukturen.

NoSQL Datenbanken geben folgendes auf:

- Relationales Schema
- Standardisierte Abfragesprache
- Beständigkeit

aber bekommen dafür im Austausch:

- Skalierbarkeit
- Flexibilität

Somit sind NoSQL Datenbanken flexibler(flexibles Datenmodell) und besser skalierbar.

15.2.1 Nicht-Relationale Key-Value Datenbanken

Datenstruktur sind assoziative Datenfelder (Schlüssel-Wert Paare). Einfache API: Lesen, Schreiben und Löschen. Key ist i.d.R. ein eindeutiger Integer oder ein eindeutiger String. Jeder Datentyp ist möglich. Als Wert können sowohl Basistypen als auch Strukturen, wie bspw. XML oder JSON gespeichert werden. Es ist Schemafrei! Daten werden als flache Sammlung ohne Struktur zwischen den Datensätzen gespeichert. Daten die zusammen gehören, werden häufig als ein Wert gespeichert. Eignet sich für Daten, die als Key/Value Paar aufgebaut sind, wie bspw. das Session Handling.

- Hohe Lese- und Schreibraten für große, unstrukturierte Datensätze
 - Einfache Verteilung über mehrere Server
 - Verwaltung im Arbeitsspeicher
- Einfaches Datenmodell
 - Zugriff & Speicherung über Schlüssel
 - Keine Joins
 - Schemafrei
- Ungeeignet für komplexe Anfragen und Aggregationen
 - Anfragen nur über Schlüssel
 - Keine standardisierten Abfragen

15.2.2 Nicht-Relationale Dokumentenorientierte Datenbanken

Sonderform der Key-Value Datenbanken. Werte sind Dokumente. Dokument wird durch einen eindeutigen Schlüssel referenziert (Zugriff erfolgt nicht zwangsläufig über Schlüssel). Dokumente bezeichnen an der Stelle eine strukturierte Zusammenstellung von Daten, bspw. JSON, XML, YAML, Dokumente sind in sogenannten Collections zusammengefasst. Hohe Flexibilität bezüglich der Struktur einzelner Datensätze. Effiziente Verarbeitung beliebiger verschachtelter Daten. Vorteilhaft wenn Struktur im Vorfeld nicht bekannt ist oder sich dynamisch ändert. Schemafrei und skalierbar. Aktualisierung des gesamten Dokuments oder einzelner Attribute.

- Vorteile:
 - Geringe Latenzzeiten
 - Tiefe Strukturen
 - ungleiche Strukturen
- Nachteile:
 - in der Kontrolle ob falsche Daten in die Datenbank kommen, da keine starre Struktur

15.2.3 Nicht-Relationale Graphendatenbanken

Daten werden in Form von Graphen gespeichert. Knoten die durch Kanten miteinander verbunden sind. Knoten repräsentieren einzelne Datensätze und Kanten sind Beziehungen. Kanten und Knoten können Eigenschaften (Properties) in Form von Key/Value Paaren haben. Einzelne Knoten sind vergleichbar mit Dokumenten in dokumentenorientierten Datenbanken.

Geeignet wenn man schnell die Beziehungen zwischen einzelnen Datensätzen erfahren möchte

15.2.4 Nicht-Relationale Spaltendatenbanken

Bzw. Wide Column Store, Column Family Store, Tabular Data Store. Tabellenartige Struktur mit flexiblem relationalen Schema. Skalierbar. Geringe Latenzzeiten über Lese- und Schreibzugriffe. Unterstützung von Milliarden Zeilen und Millionen von Spalten. Keine Joins und Transaktionen über mehrere Zeilen. Indexiert nach Zeilenschlüssel und Spaltenschlüssel. Ermöglicht schnellen, direkten Zugriff. Speicherung der Daten z.B. in lexikographischer Reihenfolge der Zeilenschlüssel (hängt von der Implementierung ab). Mehrere Versionen pro Zelle möglich.

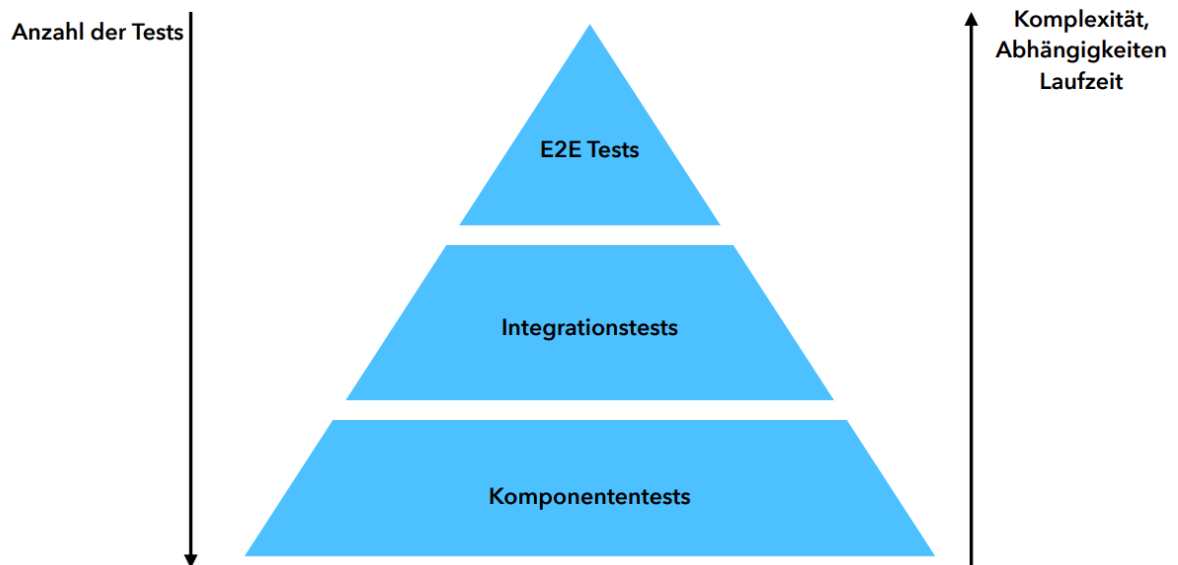
16 Testing

Je komplexer die Anwendung, desto schwieriger bei Änderungen vom Quellcode die ungewünschten Nebenwirkungen abzuschätzen. Sicherstellen, dass der neue Code keine Bugs enthält und noch funktioniert. Das Testing kann sowohl für das Frontend- als auch das Backend relevant werden. Heutzutage meistens automatisierte Tests. In der Praxis komplex, da Funktionen teilweise auf andere Datenquellen beruhen und auch komplexe Rückgabewerte erzeugen.

Arten von Tests:

- Komponententests
- Integrationstests
- End-to-End Tests
- Kompatibilitätstests
- Performancetests
- Sicherheitstests

Testing - Die Testpyramide



16.1 Test Driven Development - TDD

Test Driven Development (deutsch: testgeleitete Entwicklung; kurz: TDD) ist ein Entwicklungs- und Designparadigma für Software, bei dem das Testen von Programmkomponenten dazu verwendet wird, den gesamten Prozess der Softwareentwicklung zu leiten. Zuerst werden Tests geschrieben und dann die Funktion welche getestet wird.

Vorteile:

- Saubere Schnittstellen
- Klar gekapselte Funktionalität
- Bessere Testbarkeit

Vorgehen:

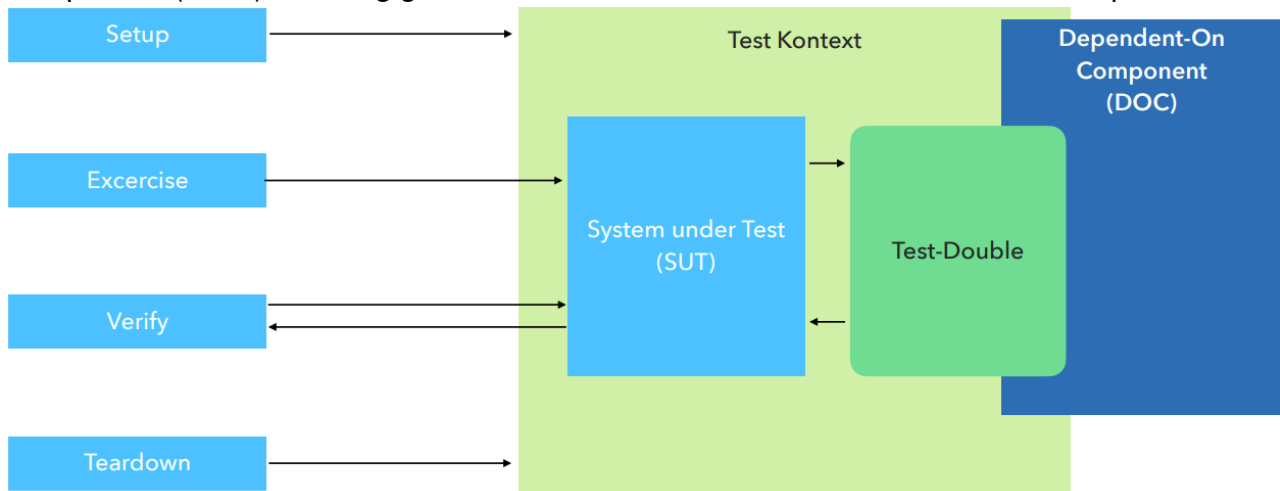
1. Schreiben des Tests
2. Ausführen des Tests
3. Implementierung der Funktionalität
4. Ausführen des Tests
5. (Refactoring)

16.2 Testabdeckung

Die Testabdeckung gibt Auskunft darüber, wieviel Code durch Tests durchlaufen wurde. Es gibt spezielle Anwendungen, die bei der Ausführung der Testfälle explizit aufzeichnen, welche Code Teile durchlaufen wurden. Je mehr Code, desto besser. Insbesondere Grenzfälle können so entdeckt werden.

16.3 Test-Doubles

Tests sollten möglichst keine externen Abhängigkeiten haben. In der Praxis nicht immer möglich, z.B. Datenbankzugriffe. Komponenten in Abhängigkeit heißen auch **Dependent-On Component (DOC)**. Abhängigkeiten erschweren das isolierte Testen einer Komponente.



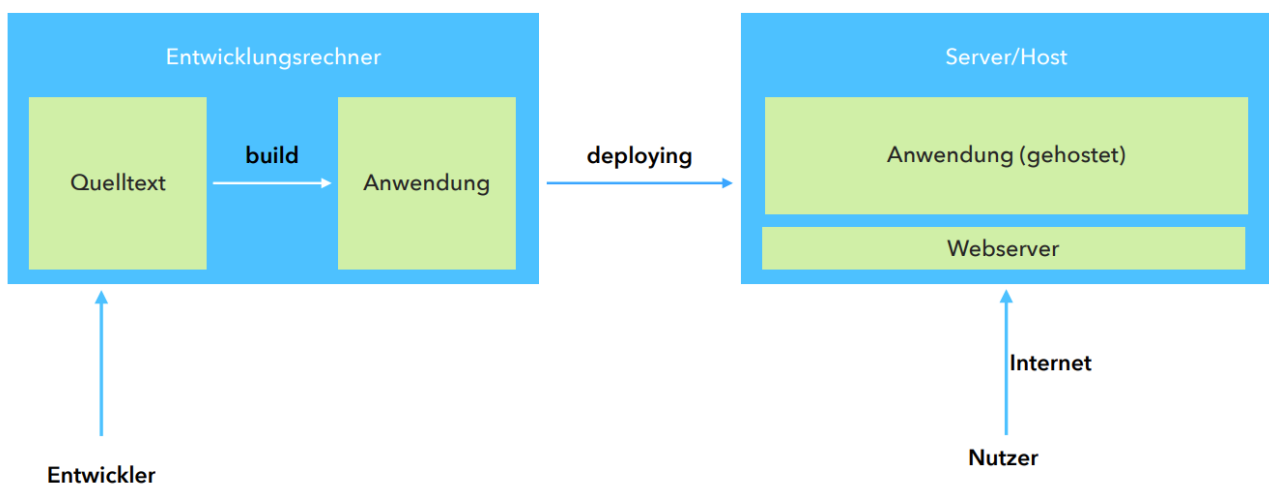
Teilen sich auf in Spies, Stubs und Mocks. Spies werden eingesetzt wenn kein lesender Zugriff auf eine Komponente möglich ist. Diese werden dann anstatt der nicht ansprechbaren Komponente angesprochen und zählen wie oft sie aufgerufen werden. Stubs simulieren eine andere Komponente. Mocks sind ähnlich wie Spies, jedoch erfolgt die Verifikation innerhalb des Mock-Objektes.

17 Deployment and Hosting

Build: Bauen einer Webanwendung für den Produkteinsatz

Deploy: Installieren der gebauten Webanwendung auf dem Server

Hosting: Bereitstellen der Anwendung auf dem Server



Das Deployment wird oft durch Continuous Integration automatisiert oder in Containern(Docker) ausgeführt.

Cloud Hosting ermöglicht es eine Webanwendung über mehrere Server verteilt zu hosten. Das Load Balancing verteilt dann Anfragen gleichmäßig auf die verschiedenen Server. Anfragen werden nicht an einen bestimmten Server gestellt sondern es findet ein Auswahlverfahren statt. Dynamisches Load Balancing beachtet hierbei noch die Staten der einzelnen

Server und entscheidet sich basierend auf diesen Informationen an welchen er sich wendet. Ein Reverse Proxy ist ein Proxy in einem Rechnernetz, der Ressourcen für einen externen Client von einem oder mehreren internen Servern holt. Die Umsetzung der Adresse ist atypisch und der Richtung des Aufrufes entgegengesetzt.

- Vorteile von Cloud Computing
 - Hohe Ausfallsicherheit
 - Keine Vorhaltung redundanter Hardware
 - Produkt kann skalieren
 - Umsetzung von "Big Data" Projekten
- Nachteile
 - oftmals komplexes Abrechnungsmodell
 - Projekt muss für Cloud Dienste optimiert werden
 - Es ist nicht exakt klar, wo die Anwendung läuft -> Thema Datenschutz

Neben Cloud Hosting gibt es auch noch Shared Hosting(Mehrere Seite auf einem Webserver), VPS Hosting(Virtueller mietbarer Server - mehrere virtuelle Server auf einem physischen Server die vermietet werden) und Dediziertes Hosting(Kunde kann gesamten Server mieten).

18 Sicherheit

18.1 Injeciton

Injection: Das Einschleusen von gefährlichem Code. Schädlicher Code der vom Client an den Server gesendet und vom Server ausgeführt wird. Lücke entsteht durch schlechtes (oder nicht) validieren der Anfrage vom Client zum Server. Häufig auch SQL-Injections.

18.2 Broken Authentication

Fehlerhafte Authentifizierung. Falsche Implementierung eines Session Handlings. Kennwörter oder Session Tokens kompromittieren um so eine andere Identität anzunehmen.

18.3 Sensitive Data Exposure

Darstellung sensibler Daten. Unverschlüsselte Übertragung von Daten im Web. Senden von sicherheitsrelevanten Daten in URL. Senden und speichern von Daten nur, wenn diese im jeweiligen Applikationskontext benötigt werden.

18.4 XML External Entities

Attacken über externe XML Entitäten. Inhalt von XML Entitäten durch Referenzen auf andere XML Dateien einbinden (XML External Entity). Vorbeugung: Deaktivieren der entsprechenden Funktionalität in den XML Parsern

18.5 Broken Access Control

Defekte Zugriffskontrolle. Fehlerhafte Implementierung welche Möglichkeiten und Rechte ein Benutzer im System hat. (Bspw. Zugriff auf Ressourcen auf die nur eine Adminrolle zugreifen können sollte).

Vorbeugen mit typischen Authorization Models:

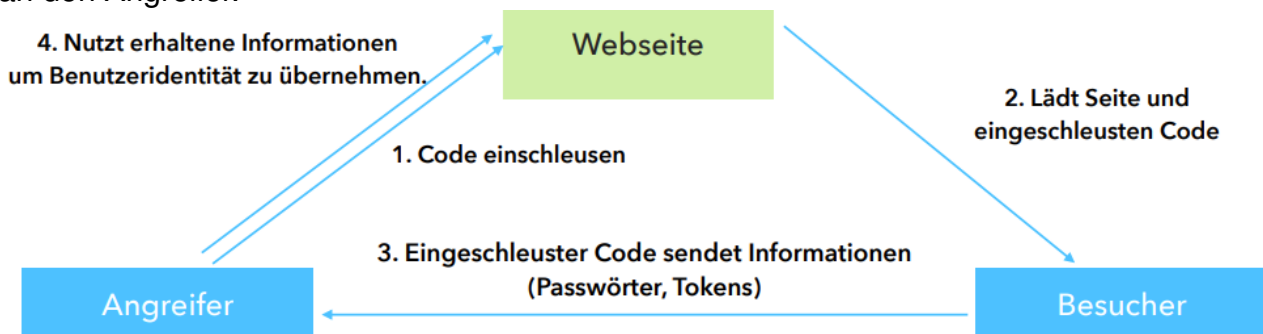
- Access Control List (ACL): Eine definierte Liste die festlegt, welche Aktionen der Nutzer auf welche Ressource durchführen darf
- Role-based Access Control (RBAC): Rechte werden Rollen zugeordnet, die man dann einem Nutzer zuweist. Nutzer können dann oftmals Gruppen zugeordnet werden, denen man ebenfalls Rollen zuweisen kann.

18.6 Security Misconfiguration

Fehlkonfiguration der Sicherheitseinstellungen. Fehlermeldung an Nutzer, die Detailinformationen verraten.

18.7 Cross-Site Scripting

Einschleusen von schädlichem Code. Einfügen von schädlichem (Java-Script) Code in eine Webseite. Schädlicher Code wird dann beim Besucher ausgeführt und sendet Informationen an den Angreifer.



Vorbeugung:

Festlegen welcher Code auf einer Webseite ausgeführt werden darf und welcher nicht. Content Security Policy (CSP).

18.8 Insecure Deserialization

Unsichere Deserialisierung. Deserialisierung: Das Erzeugen eines Objekts aus einer Folge Bytes. Hier muss auf Manipulationsmöglichkeit beachtet werden. Vorbeugen durch digitale Signaturen bei Im- und Exports.

18.9 Using Components with Known Vulnerabilities

Verwendung von Komponenten mit bekannten Sicherheitslücken. Verwenden von externen Komponenten bei denen Sicherheitslücken vorhanden sind/auftreten. Vorbeugen durch:

- Regelmäßiges updaten der Fremdkomponenten
- Nutzen von automatisierten Tools um Abhängigkeiten auf Updates zu prüfen

18.10 Unsufficient Logging & Monitoring

Unzureichende Protokollierung und Überwachung. Häufig lassen sich Hackingversuche durch ein effizientes Logging verhindern. Serverseitige Protokolle mit ausreichendem Kontext (Verdächtige Aktionen von Nutzern).