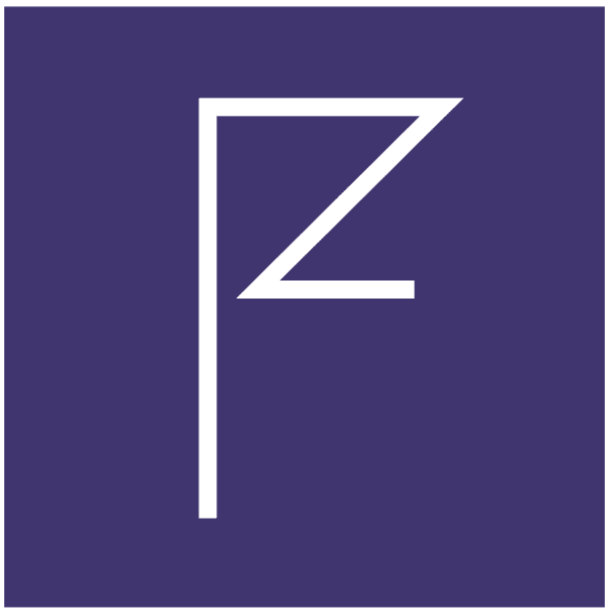

FAKE INC.

Fake Inc. Online-Shop

M239: Internetserver in Betrieb nehmen

Autor	Nico Arquint
Firma	Migros-Genossenschafts-Bund
Schule	Technische Berufsschule Zürich – TBZ
Erstellungsdatum	13.05.2019
Druckdatum	28.06.2019

Inhalt

Revisionshistorie

Datum	Version	Autor	Änderung
13.05.2019	1.0	N. Arquint	Dokument erstellt
20.05.2019	1.1	N. Arquint	*Ausgangslage* und *Anforderungen* fertig, *Infrastruktur* angefangen
27.05.2019	1.2	N. Arquint	*Infrastruktur* fertiggestellt, *Protokolle* angefangen
03.05.2019	1.3	N. Arquint	Kapitelnummerierung überarbeitet Weiterarbeit an den *Protokollen*
14.06.2019	1.4	N. Arquint	Fehlende (Unter-)Themen hinzugefügt (Infrastruktur / Protokolle)
15.06.2019	1.5	N. Arquint	*Infrastruktur* vervollständigt
16.06.2019	1.6	N. Arquint	*Protokolle* ergänzt und fortgesetzt
17.06.2019	1.7	N. Arquint	Anpassungen an diversen Themen
19.06.2019	2.0	N. Arquint	Überarbeitung der kompletten Dokumentstruktur Überarbeitung aller Texte (Rechtschreibung, etc.) Abbildungs-/Tabellenverzeichnis hinzugefügt
20.06.2019	2.1	N. Arquint	Überarbeitung aller Texte (Rechtschreibung, etc.)
21.06.2019	2.2	N. Arquint	Erarbeitung Mail-Protokolle
23.06.2019	2.3	N. Arquint	Ergänzungen Virtual Hosting / Proxy, Tests definiert
24.06.2019	2.4	N. Arquint	Deckblatt überarbeitet, Ergänzung Zertifikat Theorie
28.06.2019	2.5	N. Arquint	Fertigstellung der Dokumentation

1 Ausgangslage

Fake Inc. ist Startup-Unternehmen, welches sowohl selbstproduzierte als auch Marken-Kleider verkauft. Sie besitzen in der Schweiz drei Niederlassungen: Zürich, Bern und Genf. Wobei in Zürich gleichzeitig auch der Hauptsitz ist. Die KMU besteht aus 15 Mitarbeiter. Jeweils vier Mitarbeiter pro Filiale und drei weitere in Zürich im Büro.

Da die Firma ziemlich jung ist, gibt es noch keine Webseite, eigene E-Mails oder ähnliches. Die Kassen in den Filialen werden mit *SumUp* [1] betrieben. Zudem verläuft die Kommunikation meistens über *Threema Work*[2], oder über ein Gratis-E-Mail-Anbieter. Dafür steht in jeder Filiale ein Computer zur Verfügung.

Mittlerweile sind die Filialen bereits ein wenig bekannter geworden und die Firma möchte technisch aufrüsten. Für den Anfang benötigen sie einen Online-Shop und professionelle E-Mail-Adressen (mit eigener Domäne). Zu einem späteren Zeitpunkt wird wahrscheinlich auch noch zusätzlicher Speicher hinzukommen.

Da es sich hierbei immer noch um ein kleines Unternehmen handelt, sollte die neue Infrastruktur nicht allzu teuer sein.

[1] *SumUp Payemtns Limited* ist ein Zahlungsdienstleister mit dem Schwerpunkt auf *Mobile Point of Sale.*
[2] *Threema Work* ist ein Instant-Messanging-Dienst speziell für Unternehmen ausgerichtet.

2 Anforderungen

2.1 Lastprofil

Das Lastprofil wurde für den Sommer erstellt. Die Filialen haben pro Tag durchschnittlich 350 bis 400 Kunden pro Tag. Mit dem neuen Online-Shop werden, wenn es sehr gut läuft 100 gleichzeitige Besucher erwartet und insgesamt ca. 550–570 Personen pro Tag. Aber realistisch gesehen, werden es eher maximal 65 gleichzeitige Besucher sein, mit insgesamt 300 Kunden pro Tag.

Man erwartet hauptsächlich zu Mittags- und Feierabendzeiten eine hohe Anzahl an Usern, da viele unter der Woche arbeiten müssen. Dabei werden um die Mittagzeit mit den meisten Besucher gerechnet.

Am Wochenende wird davon ausgegangen, dass sich Samstag ähnlich wie unter der Woche verhält. Nur sinkt es zwischen Mittag und Abend nicht so stark, da die meisten Personen nicht arbeiten müssen. Am Sonntag wird es komplett anders erwartet. Oftmals schlafen die Menschen am Sonntag aus. Deswegen wird damit gerechnet, dass der Online-Shop eher am Nachmittag aufgerufen wird, statt bereits um 12.00 Uhr.

Sollte der Online-Shop doch nicht so gut ankommen, werden wenn überhaupt 25 gleichzeitige potenzielle Kunden erwartet, mit maximal 110 Besucher pro Tag.

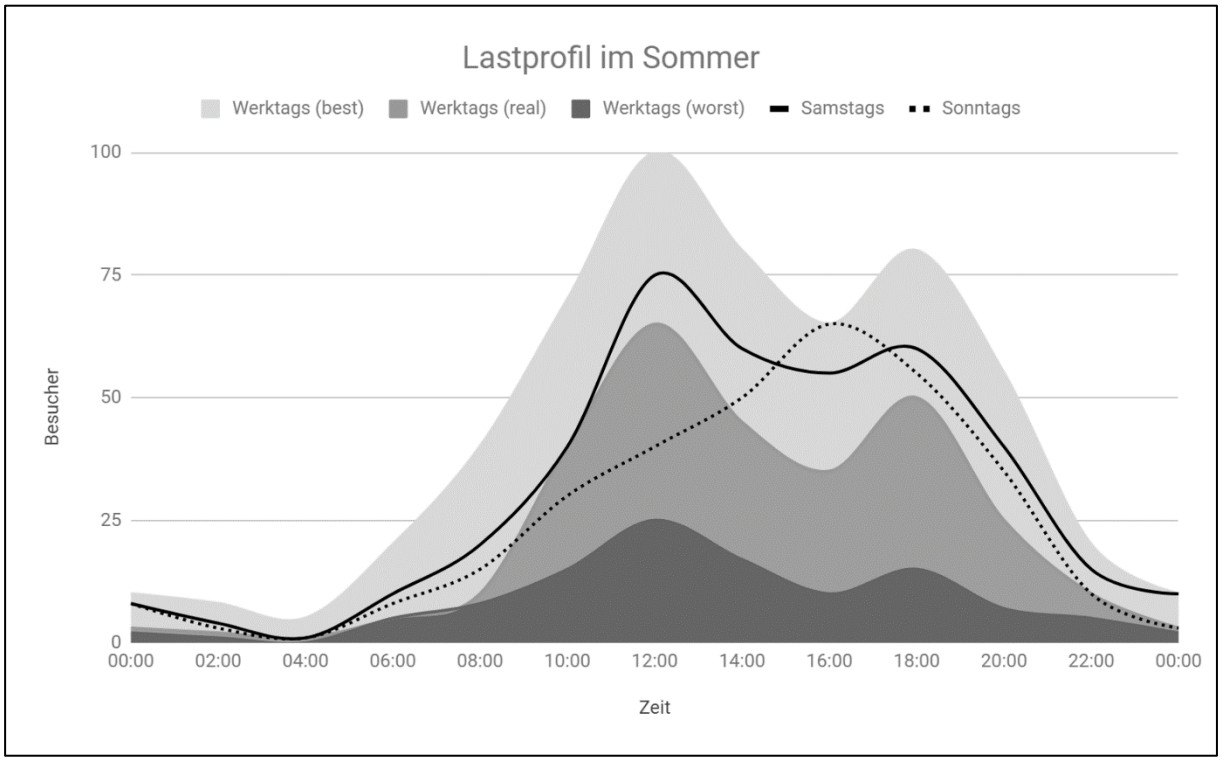

Abbildung 1: Geschätztes Lastprofil des Online-Shops im Sommer

2.2 Datenvolumen

Um die benötigten Datenvolumen zu bestimmen, wurden Messungen auf der Online-Shop Seite von *Zalando SE* (Link: https://zalando.ch) durchgeführt.

2.2.1 Gemessener Ressourcenverbrauch

Datenverkehr

Gemäss den Messungen benötigt ein Seitenaufruf ca. 10 MB. Somit würden 100 gleichzeitige Besucher pro Seite ca. 1'000 MB / 1 GB an Datenverkehr generieren. Wird das mit dem Lastprofil abgeglichen, erhält man einen Traffic von **6 GB pro Tag** (Lastprofil → 5.6 GB).

RAM

Während ein Besucher die Seite erkundigt, werden dabei ungefähr 50–80 MB RAM serverseitig genutzt (JavaScript). Auf 100 Personen hochgerechnet sind das 5'000–8'000 MB / **5–8 GB**. Da der Speicher beim Verlassen der Seite wieder freigegeben wird, muss der Arbeitsspeicher nicht auf den Tag gerechnet werden. Zusätzlich benötigt aber der Server ebenfalls noch Arbeitsspeicher, um angemessen zu laufen. Das Linux-Betriebssystem *Ubuntu Server* benötigt mindestens 1 GB, optimal wären bis zu **4 GB**, so hat man notfalls auch noch einen Puffer.

2.2.2 Geschätzter Ressourcenverbrauch

CPU

Je nachdem wie viel Scripts ablaufen, Anfragen eingehen oder Daten verarbeiten werden, benötigt es mehr oder weniger Prozessorleistung. Für eine kleine Seite braucht es mindestens **2 Kerne**, optimalerweise **4 Kerne**.

Speicher

Je nach Betriebssystem wird mehr oder weniger Speicher dafür benötigt. Viele Webserver basieren auf Linux, beispielsweise *Ubuntu Server*. Dieser empfiehlt einen Speicher von mind. **4 GB**. Für die Applikationen (Webserver, CMS, DB, Proxy, etc.) wird geschätzt bis zu **20 GB** benötigt.

Nicht zuletzt, gibt es in einem Online-Shop vor allem Bilder. Diese sollten nicht allzu viel Speicher verbrauchen, da sie maximal 100 KB gross sein sollten. Ansonsten kommt es zu lange Ladezeiten. Wenn man nun 100 Artikel mit je 5 Bilder hat, ergibt das ein Speicherbedarf von nur **50 MB**.

2.2.3 Fazit

Zusammengefasst benötigt es für einen Online-Shop folgende Ressourcen.

CPU:	mind. 2 Kerne / optimalerweise 4 Kerne
RAM:	mind. 12 GB
Speicher:	mind. 50 GB
Datenverkehr:	mind. 6 GB pro Tag → 1860 GB pro Monat (31 Tage)

Je nach Hosting können einige Ressourcen gar nicht bestimmt werden, wie zum Beispiel CPU und RAM bei einem *Webspace* (siehe 3.2.1). Aber praktisch mit allen Optionen erhält man genügend Speicherplatz, dabei hat sich vor allem 100 GB HDD-Speicher als Standard durchgesetzt.

2.3 Verfügbarkeit

Da es sich um einen Online-Shop handelt, müssen sowohl Webserver als auch der E-Mail-Server dauerhaft verfügbar sein (24/7), ausgeschlossen der Wartungsarbeiten. Werden die Server bei einem Provider gehostet, muss dieser die Verfügbarkeit (der Hardware und Infrastruktur) gewährleisten. Je nach Provider und Hosting-Option, kann die Verfügbarkeit zwischen 99.5–99.999% variieren.

2.4 Applikationen

Je nach Hosting kann sich die Applikationen variieren (SaaS / PaaS / IaaS). Aber in allen Situationen benötigt es folgende Software: Webserver mit CMS, Datenbank (z. B. MySQL), Reverse Proxy und ein Mail-Server.

2.5 Sicherheit

Bei einem Online-Shop wird sowohl mit persönlichen Daten als auch mit Zahlungsdaten gearbeitet. Dieser müssen unbedingt vor fremden Zugang abgesichert werden. Bei so einer Website stehen der Datenschutz und die Datensicherheit an erster Stelle.

3 Infrastruktur

3.1 Bestehende Infrastruktur

In allen drei Standorten werden die Kassen von einem Drittanbieter verwaltet. Dabei verläuft die ganze Kassenabrechnung digital, durch den Drittanbieter. Ebenfalls steht in jeder Filiale ein Computer. Die Kommunikation zwischen den Filialen und dem Hauptsitz verläuft gelegentlich über E-Mail und hauptsächlich über das Smartphone (Threema Work).

3.2 Neue Infrastruktur

Da Fake Inc. nur eine kleine Firma ist, macht es wenig Sinn einen eigenen Server zu betreiben. Deshalb wird auf ein Hosted Service bei einem vorher definierten Provider gesetzt.

Es gibt verschiedene Arten eine Website zu hosten:

3.2.1 Webspace

Webspaces sind nichts anderes als «Webspeicher», sprich Speicherplatz auf einem Server. Sie bilden das günstigste Modell im Web-Hosting, sind aber dementsprechend eingeschränkt.

Grundsätzlich werden *Webspaces* mit Speicherplatz, Traffic und eventuelle Sonderfunktionen geworben. Normalerweise erhält man Speicherplatz, unlimitierter Traffic («Fair-Use»), eine Anzahl an E-Mail-Adressen, ein paar Datenbanken (oftmals MySQL), evtl. ein Zertifikat und vielleicht ein paar Sonderfunktionen (Site-Builder/CMS, etc.).

Abstriche muss man bei der Rechenleistung machen. In den meisten Fällen wird die CPU-Leistung oder die RAM Menge nicht erwähnt. Diese Sachen werden im Hintergrund, bis zu einer Grenze, dynamisch zugewiesen).

Zudem ist man praktisch vom Provider abhängig, da gerne mal proprietäre Anwendungen im Hintergrund zum Einsatz kommen.

3.2.2 Virtueller Server (vServer)

Virtuelle Server (kurz *vServer*) sind mit einem Hypervisor virtualisierte Server. Dabei werden Prozessor und Arbeitsspeicher virtualisiert/geteilt (zugeteilter RAM wird nicht geteilt), hingegen der Speicherplatz dediziert ist.

Mit einem *vServer* ist man viel freier als bei einem Webspace. Dafür zahlt man natürlich auch mehr. Zudem kann die Anzahl *vCores*, *RAM* und *Speicherplatz* frei gewählt werden. Im Gegensatz zum Webspace wird bei *vServer* nur die Infrastruktur vom Provider betreut. Alles andere muss der Endkunde machen, dazu gehört auch das Betriebssystem.

Man spricht hierbei von einem PaaS (Platform as a Service). Das bedeute, dass die Infrastruktur und die Hardware vom Provider betreut wird und man eine fertige Plattform erhält. Darauf kann man sein Betriebssystem, Software, etc. aufsetzen.

3.2.3 Dedicated Server / Root-Server

Die wahrscheinlich teuerste Variante ist ein *dedizierter Server* (engl. *Dedicated Server*), auch *Root-Server* genannt (nicht zu verwechseln mit den DNS Root-Servern). Wie der Name verrät, erhält man damit einen kompletten Server vom Provider. Alle Ressourcen sind dediziert und werden nicht mit anderen geteilt.

Hier gibt es sowohl die PaaS-, als auch die IaaS-Variante. Wobei PaaS öfters vorkommt. Bei der IaaS-Version hat man einen eigenen Server verbaut, welches lediglich im Data Center angeschlossen wurde.

Mit einem *Dedicated Server* hat man unter den «statischen» Angeboten, die grösste Freiheit. Man ist nicht vom Provider abhängig und man kann/muss alles selbst machen. Es gibt, ausser beim Traffic keine Restriktionen. Solche Server kosten aber dementsprechend auch viel Geld. Nach einem *Dedicated Server* kommt nur noch der *Self-hosted Server,* bei dem der Server im eigenen Gebäude steht.

3.2.4 Cloud

Eine andere Variante ist die Cloud. Dabei gibt es sehr grosse Cloud-Anbietern wie *Google Cloud Platform*, *Microsoft Azure* und *Amazon Web Services (AWS)*. Auf solche Cloud-Plattform werden sogenannte *Instanzen* ausgeführt. Dabei handelt es sich um einzelne Services (z. B. Container), oder virtuelle Server. Auch die Ressourcen dieser *Instanzen* können beliebig ausgewählt werden.

Solch eine Plattform kann sowohl als PaaS, aber auch als IaaS verwendet werden (je nach Provider). Im Gegensatz zu den anderen Varianten, arbeitet die Cloud dynamisch. Sprich, Ressourcen können jederzeit hoch-, aber auch herunterskaliert werden. Logisch können die vorher genannten Versionen auch in CPU, RAM, Speicher und Traffic skaliert werden (vertikale Skalierung). Aber man kann schwer aus einer Maschine, zwei machen (horizontale Skalierung). In der Cloud funktioniert das problemlos. Wird einmal mehr Leistung benötigt, kann sowohl vertikal als auch horizontal erweitert werden.

Wie beschrieben, ist der grösste Vorteil die Skalierbarkeit. Und da immer mehr in die Cloud migriert wird, ist es zudem ziemlich zukunftssicher. Der grösste Nachteil aber, ist die Komplexität und je nach Provider auch die (intransparente) Kostenmodelle.

3.2.5 Fazit

Für eine kleine Firma ist ein Webspace oder ggf. eine Cloud-Lösung der beste Ansatz.

Webspace	**Cloud**
+ Einfach	+ Zukunftssicher
+ Preisgünstig	+ Skalierbar
+ Das «Sorglos»-Paket (alles vorinstalliert)	+ Freie Auswahl von Technologien
– Eingeschränkte Konfigurationen	– Komplexität
– Vom Provider abhängig	– Vom Provider abhängig
– Eingeschränkte Technologien	– Komplizierte Kostenmodell

Tabelle 1: Vergleich zwischen Webspace und Cloud

Es gibt auf beiden Seiten Vor- und Nachteile. Schlussendlich muss sich die Firma entscheiden. Je nach dem um was es sich für ein Unternehmen handelt, eignet sich eine Variante besser als die anderen. Für ein Kleinunternehmen, welches zum ersten Mal eine Website betreut, ist ein Webspace wahrscheinlich die beste Variante. Denn meisten ist bereits alles vorinstalliert und man muss lediglich seine Webseite einrichten.

4 Protokolle

4.1 DNS-Protokoll

4.1.1 Funktionalität

Das DNS-Protokoll (Domain Name System) wird grundsätzlich zur Namensauflösung benötigt. Sprich, es übersetzt Domains in IP-Adressen und umgekehrt. Dabei wird immer von rechts nach links aufgelöst.

Für die Auflösung reicht aber ein DNS-Server nicht aus. Es benötigt mindestens 3 Server um die IP-Adresse einer Domäne (z. B. `example.com`) zu erhalten. Denn ein Server kann nicht alle IP-Adressen und Domains indexieren. Deswegen gibt es die drei Server auf verschiedenen Stufen. Diese lösen die Domain immer nur zum Teil auf.

Root-Server

Weltweit gibt es nur 13 Root-Server. Sie bilden den Anfang der Namensauflösung («Root», engl. «Wurzel»). Sie beinhalten alle IP-Adressen der TLD-Server (z. B. vom, `.com`-Server).

TLD-Server

TLD bedeutet Top Level Domain und definiert die oberste Stufe einer Domain. Das sind jeweils die Endungen bei einer Domain: .com, .net, .ch, etc. Auf diesen Servern sind die IP-Adressen der Nameserver der jeweiligen Domain (z. B. Nameserver von `example.com`)

Nameserver

Die letzte Stufe bilden die Nameserver. Sie beinhalten alle IP-Adressen einer Domain und deren Sub-Domains. Darin befindet sich schlussendlich die effektive IP-Adresse eines Webservers (z. B. IP von `example.com`).

DNS-Server

Der DNS-Server ist meist lokal und führt die Namensauflösungen durch. Sprich, wird eine Website angefordert (z. B. `example.com`) wird die Anfrage an den DNS-Server gesendet. Sollte die angeforderte Domain nicht im Zwischenspeicher liegen, macht der DNS-Server (nicht der Client) die Anfragen an den Root-, TLD- und Name-Server. Am Schluss übergibt der DNS-Server die IP-Adresse an den Client.

Lokaler Cache

Eine Stufe vorher ist der lokale Zwischenspeicher eines Computers. Alle angefragten Domains werden nicht nur auf dem DNS-Server, sondern auch lokal zwischengespeichert. Dieser wird im Normalfall beim Herunter-/Hochfahren gelöscht.

Also sollte eine IP-Adresse bereits im lokalen Cache vorhanden sein, wird keine Anfrage an den DNS gesendet.

hosts Datei

Die unterste Stufe einer Namensauflösung bildet die sogenannte `hosts`-Datei. Jedes Betriebssystem hat eine solche Datei. Diese Datei ist persistent und wird nicht gelöscht. Darin können manuelle IP-Domain-Zuweisungen hinterlegt werden.

Sollte also eine IP/Domain bereits in der `hosts`-Datei vorhanden sein, wird der lokale Cache nicht mehr überprüft. Stattdessen wird direkt die IP-Adresse in dieser Datei verwendet.

4.1.2 FQDN

FQDN (**F**ully-**Q**ualified **D**omain **N**ame) beschreibt den vollständigen Namen einer Domain/Hosts und ist gleichzeitig eine absolute Adresse.

Beispiel:

Domain	`www.example.com.`			
FQDN	`www`	`.example`	`.com`	`.`
Schema	`3rd-Level`	`.2nd-Level`	`.Top-Level`	`.Root-Label`

Tabelle 2: Beispiel einer FQDN

Man beachtet den Punkt am Schluss der Domain. Da nach der Top-Level das Root-Label kommt, wird dieser ebenfalls durch einen Punkt abgetrennt. Aber da das Root-Label aus einer leeren Zeichenkette besteht, wird üblicherweise der Punkt weggelassen:

```
www.example.com
```

Wird der Punkt weggelassen, gilt die Domain theoretisch nicht mehr als eine absolute, sondern eine relative Adresse. Somit gilt sie auch nicht mehr als FQDN.
Im Browser können beide Schreibweisen verwendet werden (mit und ohne Punkt).

Ein FQDN ist nicht auf eine Anzahl Ebene beschränkt. Weitere Ebenen können problemlos hinzugefügt werden: `n-th-Level.[...].3rd-Level.2nd-Level.Top-Level.Root-Label`

4.1.3 DNS Request

Eine DNS-Abfrage für die Website `example.com` sieht ungefähr so aus (ohne `hosts` und dem lokalen Cache):

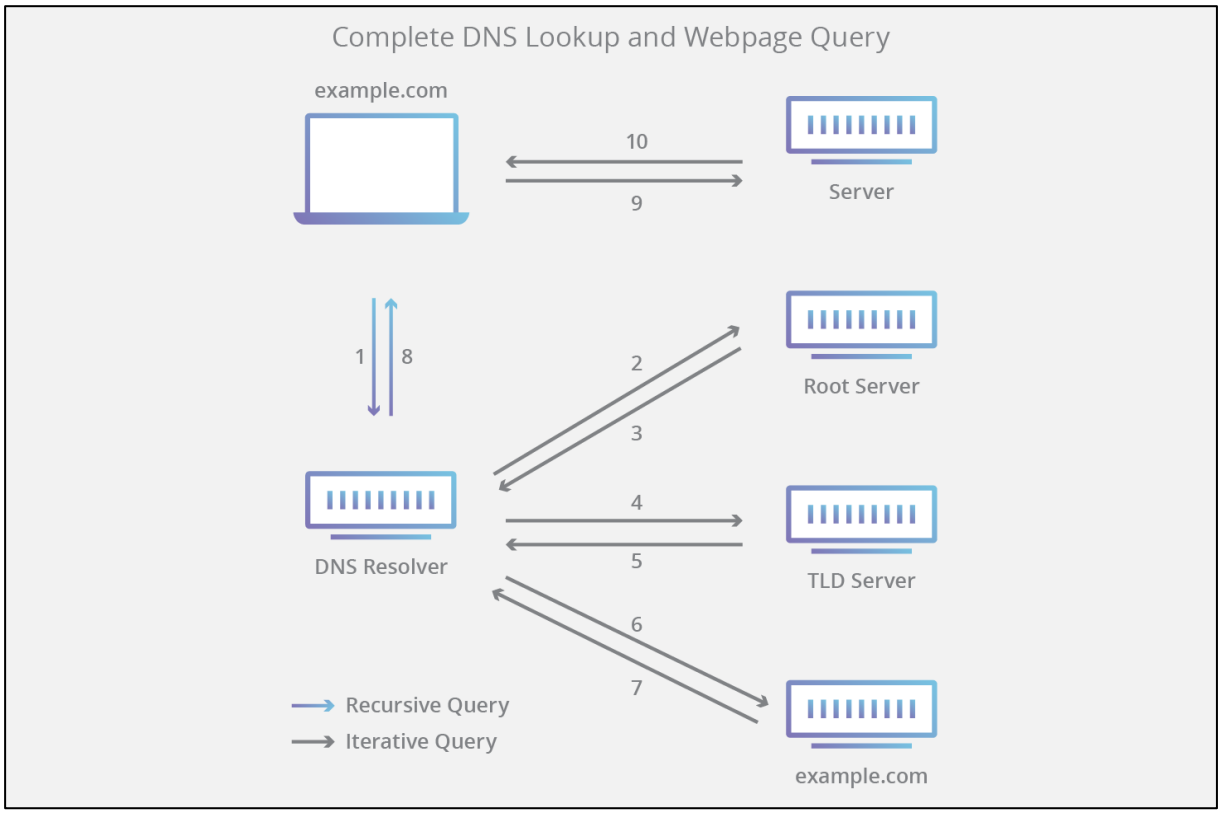

Abbildung 2: Ablauf einer DNS-Anfrage
(Quelle: https://www.cloudflare.com/learning/dns/what-is-dns/)

1. Der Client möchte die Domain example.com auflösen und macht eine Abfrage beim DNS-Server.
2. Da die IP-Adresse noch nie abgefragt wurde, muss einer der 13 Root-Server abgerufen werden.
3. Einer dieser Server antwortet mit der IP-Adresse des zuständigen TLD-Servers (`.com`).
4. Anschliessend wird der TLD-Server vom DNS-Server angefragt.
5. Dieser antwortet mit dem entsprechenden Nameserver der Domain.
6. Die letzte Abfrage findet beim Nameserver statt, dieser beinhaltet die IP der Website.
7. Der Nameserver antwortet mit der entsprechenden IP-Adresse.
8. Daraufhin leitet der DNS-Server die IP an den Client weiter.
9. Der Client verbindet sich mit dem Webserver.
10. Zum Schluss antwortet der Webserver am Client.

Dabei zu beachten ist, dass alle Abfragen (Punkt 2–8) vom DNS-Server ausgehen. Der Client fragt lediglich seinen DNS-Server an.

4.1.4 Resource Records

DNS Records, oder *Resource Records* (RR) bezeichnen die Einträge im DNS. Dabei gibt es verschiedene Darstellungen: ASCII-Darstellung, Zonendatei, DNS-Cache oder im DNS-Paket.

Das Schema eines RRs im ASCII-Format sie wie folgt aus:

```
<name> [<ttl>] [<class>] <type> [<rdlength>] <rdata>
```

Parameter	Bedeutung
`<name>`	Domain-Name des Objekts
`<ttl>`	**T**ime-**T**o-**L**ive in Sekunden. Gültigkeit des Eintrags
`<class>`	Protokollgruppe
`<type>`	Typ des Resource Records
`<rdlength>`	Länge der Daten
`<rdata>`	Resource Data, z. B. IP-Adresse bei einem A- oder NS-Record

Tabelle 3: Mögliche Parameter eines Resource Records

Für den `class`-Eintrag gibt es 4 verschiedene Klassen: IN, CH, HS und CS. Dabei wird praktisch nur IN verwendet.

Oft genutzte Record-Typen

Typ	Beschreibung	Funktion
`A`	**A**ddress Record	Gibt die IPv4-Adresse zurück. Zuordnung Domain → IPv4-Adresse.
`AAAA`	IPv6 Address Record	Dasselbe wie A-Record, aber für IPv6.
`CNAME`	Canonical Name Record	Alias für eine Domain.
`MX`	Mail Exchange Record	Definiert den Mailserver für eine Domain.
`TXT`	Text Record	Ursprünglich ein frei definierbarer Text. Heute auch für Verifizierungen genutzt.
`NS`	Name Server Record	Nameserver der Domain.
`SRV`	Service Locator Record	Definiert IP-basierte Dienste einer Domain.
`PTR`	Pointer Record	Reverse Mapping: IP → Domains

Tabelle 4: Oft genutzte Record-Typen

4.1.5 DNS Lookup

Um eine Domain manuell nachzuschlagen gibt es verschiedene Tools. Entweder verwendet man eine Website wie https://mxtoolbox.com/. Oder verwendet die Befehlszeile des Betriebssystems (Windows: CMD/PowerShell, Linux/MacOS: Terminal).

Beispielsweise gibt es auf allen Plattformen den `nslookup`-Befehl:

```
C:\>nslookup example.org
Server:  UnKnown
Address:  192.168.1.1

Nicht autorisierende Antwort:
Name:    tbz.ch
Address:  149.126.4.25
```

Listing 1: Einfache Namensauflösung via Befehlszeile mit nslookup

Mit nslookup lassen sich auch detaillierte Abfragen machen (Alle Einträge abfragen):

```
C:\>nslookup
> set type=any
> tbz.ch
Server:  UnKnown
Address:  192.168.1.1

Nicht autorisierende Antwort:
tbz.ch  internet address = 149.126.4.25
tbz.ch  MX preference = 20, mail exchanger = mx002.tam.ch
tbz.ch  MX preference = 10, mail exchanger = mx001.tam.ch
tbz.ch  nameserver = dns3.mhs.ch
tbz.ch  nameserver = dns2.mhs.ch
tbz.ch  nameserver = dns1.mhs.ch

dns1.mhs.ch     internet address = 213.160.43.203
dns2.mhs.ch     internet address = 213.188.32.64
dns3.mhs.ch     internet address = 213.188.32.84
```

Listing 2: Erweiterte Namensauflösung via Befehlszeile mit nslookup

Als Einzeiler kann der Befehl auch wie folgt ausgeführt werden:

```
C:\>nslookup -q=any tbz.ch
```

Listing 3: Einzeiler – Records-Abfrage via Befehlszeile mit nslookup

Möchte man einen spezifischen Eintrag abfragen, kann dies wie folgt durchgeführt werden:

```
C:\>nslookup
> set type=mx
> tbz.ch
Server:  UnKnown
Address:  192.168.1.1

Nicht autorisierende Antwort:
tbz.ch  MX preference = 20, mail exchanger = mx002.tam.ch
tbz.ch  MX preference = 10, mail exchanger = mx001.tam.ch

tbz.ch  nameserver = dns1.mhs.ch
tbz.ch  nameserver = dns3.mhs.ch
tbz.ch  nameserver = dns2.mhs.ch
dns1.mhs.ch     internet address = 213.160.43.203
dns2.mhs.ch     internet address = 213.188.32.64
dns3.mhs.ch     internet address = 213.188.32.84
```

Listing 4: MX-Record Abfrage in der Befehlszeile via nslookup

Auch hier kann folgender Einzeiler verwendet werden:

```
C:\>nslookup -q=mx tbz.ch
```

Listing 5: Einzeiler – MX-Record Abfrage via Befehlszeile mit nslookup

4.2 Web-Protokolle

4.2.1 URL

Die ***U**niform **R**esource **L**ocator* (kurz *URL*) wird verwendet, um Ressourcen zu identifizieren und zu lokalisieren. Einfach gesagt: Es sucht die angegebene Ressource. Es wird hauptsächlich im Browser genutzt, um Webseiten aufzurufen. Mit der URL lässt sich aber auch andere Sachen machen, als nur eine Website zu öffnen. Damit lässt sich eine FTP-Verbindung aufbauen, Mail-Client öffnen und bestimmte Person anschreiben oder ein Ordner auf einem Rechner öffnen und durchsuchen.

Dabei ist die URL in zwei Teilen unterteilt: *Schema* und *Schema-spezifischer Teil*

```
<scheme>:<scheme specific part>
```

Listing 6: URL Schema

Beispiel anhand des HTTPS-Protokolls:

```
      |------------------ Schema-spezifischer Teil ------------------|

 https://max:muster@www.example.com:8080/index.html?p1=A&p2=B#ressource
 \___/   \_/ \____/ \_____________/ \__/\_________/ \_______/ \_______/
   |      |    |           |          |       |          |         |
Schema    | Kennwort      Host       Port    Pfad       Query    Fragment
      Benutzer
```

Listing 7: Aufbau einer URL anhand eines HTTPS-Beispiels
(Quelle: https://de.wikipedia.org/wiki/Uniform_Resource_Locator)

4.2.2 HTTP

HTTP (***H**ypertext **T**ransfer **P**rotocol*) ist ein Protokoll zur Übertragung von Daten. Hauptsächlich wird es verwendet, um Webseiten (in einem Browser) anzuzeigen. Dabei wird anhand der URL (siehe oben) ein `HTTP GET` Paket an den Server gesendet. Dieser wiederum sendet eine Antwort zurück. Im Normalfall (wenn keine Probleme bestehen) wird ein `HTTP 200 OK` Code zurückgesendet, inkl. Website-Daten.

Je nachdem was für eine Methode (siehe 4.2.4) verwendet wird, beinhaltet die Anfrage verschiedene Informationen. Ein `GET` könnte wie folgt aussehen:

```
GET /objectserver/restapi/alerts/status HTTP/1.1
Accept: application/json
Authorization: Basic dGVzdHVzZXIwMTpuZXRjb29s
Host: localhost
Connection: keep-alive
```

Listing 8: HTTP Request Beispiel

Dabei wird die erste Reihe als ***Request Line*** bezeichnet. Sie beinhaltet die Methode (`GET`), die URL/Ressource (`/objectserver/restapi/alerts/status`) und die Protokoll-Version (`HTTP/1.1`).

Ab der zweiten Zeile (keine Leerzeile) folgt der ***Header***. Dieser beinhaltet diverse "Einstellungen" wie z. B. welche Dateitypen der Browser bei der Anfrage akzeptiert, welcher Host zuständig ist, ggf. Cookies, etc.

Bei gewissen Methoden kann auch noch der ***Body*** dazukommen. Dieser beinhaltet die effektiven Daten. Zum Beispiel wenn etwas hochgeladen wird und dabei die `PUT`-Methode verwendet wird. Der Body wird mit einer Leerzeile vom Header getrennt.

Nach dem Request könnte die Antwort wie folgt aussehen:

```
HTTP/1.1 200 OK
Cache-Control: no-cache
Server: libnhttpd
Date: Wed Jul 4 15:32:03 2012
Connection: Keep-Alive:
Content-Type: application/rdf+xml
Content-Length: 24860

      {
"rowset":   {
            "osname":   "NCOMS",
            "dbname":   "alerts",
            "tblname":  "status",
            "coldesc":  [{
⋮
```

Listing 9: Ausschnitt einer möglichen HTTP Response

4.2.3 HTTPS

HTTPS ist eine Abkürzung für ***HTTP** over **S**SL*. Es ist lediglich die verschlüsselte Variante von HTTP. Während dem Verbindungsaufbau (TCP Three-Way-Handshake) wird dabei die Verschlüsselung ausgehandelt. Je nach dem was der Browser/Server unterstützt, wird eine stärke oder schwächere Verschlüsselung verwendet.

4.2.4 Methoden

HTTP bietet diverse Methoden für verschiedene Aktionen (vergleichbar mit SQL Befehle):

Methode	Funktion
`GET`	Meistverwendete Methode, fordert eine Ressource an (Website anzeigen)
`POST`	Sendet unbegrenzte Menge an Daten an den Server zur Auswertung
`HEAD`	Fordert den HTTP-Header einer GET-Anfrage, ohne den Body
`PUT`	Wird benötigt, um neue Ressourcen auf dem Server zu erstellen
`PATCH`	Ändert eine vorhandene Ressource, ohne alles neu hochzuladen
`DELETE`	Löscht eine Ressource auf dem Server
`TRACE`	Der Server antwortet mit derselben Anfrage wie der Client
`OPTIONS`	Sendet eine Liste vom Server unterstützte Methoden
`CONNECT`	Wird von Proxy-Servern für SSL-Tunneling verwendet

Tabelle 5: HTTP-Methoden und deren Funktion

Um eine Website abzurufen, gibt es dafür zwei Methoden:

GET-Methode

Dabei wird alles über die URL abgefragt. Sprich die Parameter werden ebenfalls in der URL angegeben. Dabei wird nach der Ressource ein `?`-Zeichen verwendet, um die Parameter von der Ressource zu trennen. Sie werden mit dem Schema `Parameter=Wert` mitgegeben. Mehrere Wertepaare werden mit einem `&`-Zeichen verbunden.

POST-Methode

Über die `POST`-Methode können grössere Datenmengen an den Server gesendet werden, da nicht alle Informationen über die *URL* gesendet werden, sondern ein Teil im *Body* zu finden sind.

Der Aufbau des *HTTP*-Pakets ist ähnlich wie zum `GET`, nur wird die Methode `POST` verwendet und es enthält einen Body mit den zusätzlichen Informationen. So wird beispielsweise die Parameter `Parameter=Wert&Parameter2=Wert2` im *Body*, statt in der *URL* mitgegeben.

4.2.5 Cookies

Ein ***Cookie***, auch ***Magic Cookie*** genannt, sind kleine Datenpakete, welche vor allem auf Webseiten eingesetzt werden. Diese *HTTP-Cookies* sind dabei eine Unterkategorie der *Magic Cookies*.

Die *HTTP-Cookies* werden dafür genutzt, um Website-spezifische Informationen lokal auf den Clients temporär zu speichern. Diese Cookies werden als Hash gespeichert und werden bei Bedarf wieder an den Server gesendet.

Solche Cookies helfen dabei, Webseiten zu individualisieren, so werden z. B. Sprache, Design, oder andere Einstellungen gespeichert. Ebenfalls können sich Benutzer über Cookies authentifizieren, so müssen sie sich nach einmaligen einloggen, nicht erneut anmelden.

Aber Cookies werden heutzutage auch sehr gerne für Werbung eingesetzt. Indem persönliche bzw. gerätespezifische Daten gespeichert und am Server übermittelt werden.

Früher hatte man kaum Kontrolle, welche Daten eine Website über eine Person/ein Gerät gespeichert hat. Seit der Einführung mit der EU-DSGVO, müssen alle Webseiten mit Kunden aus der EU, den Benutzer die Möglichkeit geben, anzugeben was für Daten von ihnen gespeichert werden dürfen. Zudem dürfen Benutzer auch verlangen, alle seine Daten von einer Website anzusehen und gegebenenfalls zu löschen.

Da im Internet (fast) jeder, auf jede Seite zugreifen kann, müssten somit theoretisch alle Webseiten der DSGVO nachgehen.

4.2.6 Content-Type

Der *Content-Type*, eigentlich *Internet Media Type* genannt, klassifiziert unter anderem die Daten in einem HTTP Response. Fordert der Browser eine Seite auf, erhält dieser entsprechend eine Antwort. Im Header des HTTP Responses wird der Typ des Inhalts definiert. So weiss der Browser, was er mit den erhaltenen Daten anfangen soll.

Im HTTP-Header nennt sich das Feld *Content Type*, von da auch der Name. Der *Content Type* hat folgendes Schema:

```
Content-Type: <media-type>/<file-type>
```

Listing 10: Content-Type Schema

Teilweise wird dahinter auch noch der Zeichensatz mit `charset`, oder ein `boundary` angehängt.

Beispiel:
Der Browser fordert die Seite example.com auf. Als Antwort erhält er eine HTML-Datei. Dabei findet man im Header der Antwort folgende Zeile: `Content-Type: text/html`

Folgende Medientypen existieren:

Medientyp	Beschreibung
`application`	nicht-interpretierte binäre Daten, Infos für ein bestimmtes Programm
`audio`	Audiodateien; MP3, WAV, etc.
`example`	Beispiel-Medientyp für Dokumentationen
`image`	Grafiken; JPG, PNG, TIFF, etc.
`message`	Nachrichten; z. B. RFC822
`model`	Mehrdimensionale Daten
`multipart`	Mehrteilige Daten
`test`	Text; HTML, PLAIN, etc.
`video`	Videodateien; MP4, etc.

Tabelle 6: Möglich Medientypen im HTTP-Header

4.2.7 Negotiation

Als ***Negotiation*** oder ***Content Negotiation*** wird in HTTP die Verhandlung der vom Client definierten Vorzüge definiert.

Im HTTP-Header werden folgende Header akzeptiert:

Header-Feld	Beschreibung (Gegenstück im HTTP-Response)
`Accept`	Liste akzeptierter Medien-Typen (`Content-Type`)
`Accept-Charset`	Liste akzeptierter Zeichensätze (`charset` im `Content-Type`)
`Accept-Encoding`	Liste akzeptierter Codierungen (`Content-Encoding`)
`Accept-Language`	Liste akzeptierter Sprachen (`Content-Language`)

Tabelle 7: HTTP Negotiation – Mögliche Header

Das Schema für alle vier Felder entspricht:

```
Header: first, second, third, ...
Accept-Language: de-ch, de, en;q=0.5, fr;q=0.2
```

Listing 11: Accept-Header Schema

Dabei kann mit `;q=0.0` jeweils noch eine Gewichtung angehängt werden. Sollte nicht die Hauptsprache verfügbar sein, wird die Sprache mit der höchsten Gewichtung ausgewählt.

4.2.8 Status Code

Es gibt diverse HTTP Status Code, welche nach jeder Anfrage zurückgegeben wird. Je nach Umständen gibt es einen anderen Code zurück.
Die Codes sind wie folgt strukturiert:

1xx - Information

Es gibt drei 100er Codes (100-102), welche Informationen vorweisen. Zum Beispiel der Code 100 bedeutet, dass die aktuelle Anfrage noch nicht zurückgewiesen/abgebrochen wurde. So kann der Client weitere Anfragen senden.

2xx - Erfolgreich

Ist die Anfrage vom Client erfolgreich, wird ein 200er Code zurückgegeben. Standardmässig ist es der Code 200 “OK”, dabei wird das Ergebnis in der Rückmeldung übertragen. Es gibt aber auch den Code 202 “Accepted”, bei dem zwar die Anfrage akzeptiert wurde, aber das Ergebnis erst später ausgeführt wird (nicht garantiert).

3xx - Umleitung

Wird man beim Aufrufen einer Seite umgeleitet, erhält man einen 300er Code. Wie beispielsweise der Code 301 “Moved Permanently”. Diesen erhält man, wenn die Ressource dauerhaft verschoben wurde. Oftmals wird diese Variante für die HTTP zu HTTPS Weiterleitung verwendet, um eine abgesicherte Verbindung zu garantieren/erzwingen.

4xx - Client Error

Bei einer fehlerhaften Anfrage, wird unterschieden, ob der Fehler Clientseitig oder Serverseitig stattgefunden hat. Die 400er Codes lässt auf ein Client-Fehler hindeuten. Sehr berühmt ist der Code 404 “Not Found”. Dieser wird zurückgegeben, wenn die angegebene Ressource nicht gefunden werden kann, beispielsweise wenn man sich in der URL vertippt hat.

5xx - Server Error

Besteht ein Fehler serverseitig, wird ein 500er Code als Antwort gesendet. Logischerweise kann so ein Code nicht bedeutet, dass der Webserver offline ist. Vielmehr bedeutet es, dass auf dem Server etwas falsch konfiguriert wurde. So zum Beispiel der Code 503 “Service Unavailable”. Diesen erhält man, wenn der Server (temporär) nicht auf Ressource zugreifen kann. Oftmals passiert das, wenn der Webserver überlastet ist, oder wenn gerade Wartungsarbeiten laufen.

9xx - Proprietär

Die 900er Codes sind speziell, denn es handelt sich um nicht offizielle/standardisierte Codes. Dennoch werden sie teilweise von Entwickler für eigene Statuscodes verwendet.

Diese Codes sind somit vernachlässigbar.

4.2.9 Umleitungen

Umleitungen werden über die Webserver-Anwendung (z. B. Apache) eingestellt. Je nach Software wird es anders eingestellt, aber im Grunde funktioniert alles gleich. Der Redirect wird auf dem Hostname gesetzt (siehe 0). Sprich wird ein bestimmter Hostname eingegeben, wird es entsprechend umgeleitet.

Oftmals wird HTTP zu HTTPS weitergeleitet, damit die Verbindung abgesichert ist und nicht abgehört wird. Mit dem HTTP/S-Protokoll wird dabei ein 300er-Statuscode zurückgegeben.

4.3 Mail-Protokolle

Für die E-Mail-Kommunikation gibt es verschiedene Protokolle, die drei wichtigsten werden hier beschrieben.

Die Kommunikation bei einem E-Mail-Versand funktioniert wie folgt (vereinfacht):

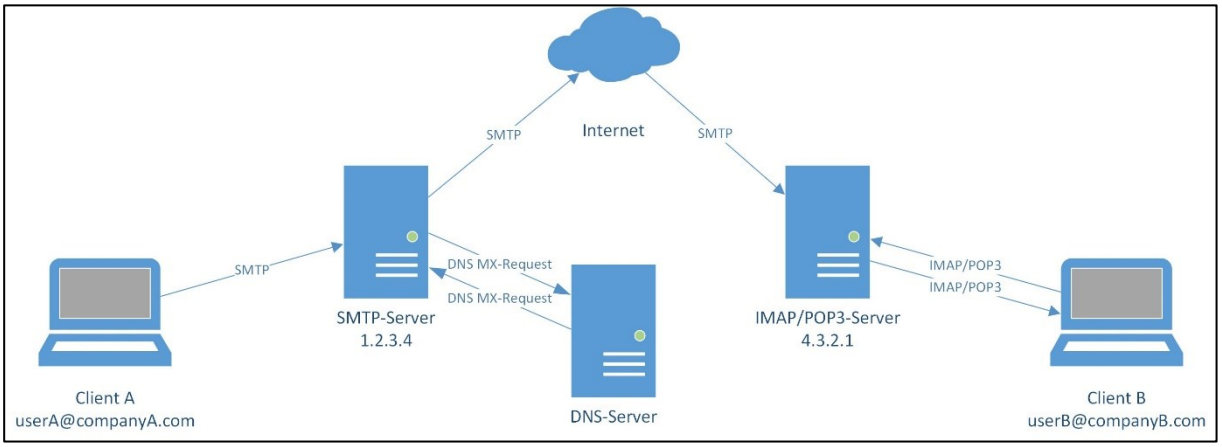

Abbildung 3: E-Mail-Kommunikation – Abfolge der Protokolle

1. User A sendet eine E-Mail an User B.
    a. Protokoll: SMTP
2. Mail wird an SMTP-Server gesendet (→ 1.2.3.4).
    a. Protokoll: SMTP
3. SMTP-Server löst mithilfe des DNS den Hostname des Empfängers auf.
    a. Protokoll DNS, MX-Record Request
4. SMTP-Server leitet die Mail zur aufgelösten IP-Adresse weiter (→ 4.3.2.1).
    a. Protokoll: SMTP
5. Mail wird auf dem IMAP/POP3-Server des Empfängers in einem «Postfach» gelagert.
    a. Protokoll: IMAP/POP3
6. E-Mail-Client (z. B. Outlook) auf Client B synchronisiert alle x Minuten mit dem IMAP/POP3-Server und lädt die neuen Nachrichten herunter.
    a. Protokoll: IMAP/POP3

4.3.1 SMTP

Für das Senden von E-Mails wird das ***S**imple **M**ail **T**ransfer **P**rotocol* (kurz: *SMTP*) verwendet. Dieses Protokoll ist praktisch für die gesamte Kommunikation zuständig. Es wird benötigt, um die Mails bis zum Postfach des Empfängers zu senden. Ab da übernimmt dann ein anderes Protokoll (siehe unten).

Das Mail-Protokoll hört standardmässig auf den Port **25/TCP**. Die abgesicherte Variante mit SSL/TLS hört auf **465/TCP** und die Variante mit MSA (*Mail Submission Agent*) und STARTTLS auf **587/TCP.**

SMTP war ursprünglich ein reines ASCII-Protokoll. Somit konnten damit keine Binärdaten übermittelt werden. Erst mit der Einführung diverser Standards, z. B. MIME (Multipurpose Internet Mail Extension) liessen sich anschliessend Binärdaten in ASCII kodieren.

Da es sich um ein textbasiertes Protokoll handelt, kann man über Telnet auch manuell eine E-Mail versenden. Ähnlich zu HTTP gibt es auch hier einfache Befehle:

Befehl	**Funktion**
`HELO`	Startet die Kommunikation, wird mit dem Sender-Server angegeben.
`EHLO`	Dasselbe wie `HELO`, nur wird Extended SMTP verwendet.
`MAIL FROM`	Definiert die Absenderadresse.
`RCPT TO`	Definiert den Empfänger, pro Empfänger muss der Befehl wiederholt werden
`SIZE`	Gibt die geschätzte Grösse der zu sendenden E-Mail an.
`DATA`	Startet die Übertragung der Mail-Daten.
`VRFY`	Lässt eine Adresse überprüfen.
`TURN`	Wechselt die Rollen des Clients und Servers, ohne eine neue Verbindung aufzubauen.
`AUTH`	Damit kann sich der Client mit User und Passwort authentifizieren.
`RSET`	Die aktuelle Übertragung wird abgebrochen/zurückgesetzt.
`EXPN`	Fragt nach einer Bestätigung der Identifikation der Mailing-Liste.
`HELP`	Zeigt einige Hilfen für die Befehle an.
`QUIT`	Beendet die Kommunikation mit dem SMTP-Server.

Tabelle 8: SMTP CLI Befehle
(Quelle: https://serversmtp.com/smtp-commands/)

4.3.2 POP3

POP steht für ***P**ost **O**ffice **P**rotocol* und ein Protokoll, welches E-Mail-Client erlaubt Mails vom Mail-Server abzuholen. Die *3* in Namen steht für die dritte Version dieses Protokolls.

POP3 hört auf den Port **110/TCP**. Mit einer abgesicherten Leitung (SSL/TLS) hört es auf **995/TCP**.

Dieses Protokoll ist bereits ziemlich alt, weswegen es auch nur eingeschränkte Funktionalität bietet. Es erlaubt lediglich das Auflisten, Abholen und Löschen von E-Mails auf dem Mail-Server. Sprich es wird nicht einmal eine lokale Kopie erstellt. Der Nachfolger bzw. das neuere Protokoll ist *IMAP* (siehe unten). Zusätzlich bietet es keine Synchronisierung zwischen verschiedene Geräte an. Wird zum Beispiel eine Mail als gelesen markiert, zeigt es das nicht auf andere Clients an.

Dafür ist keine durchgehende Verbindung mit dem Server nötig. Sie wird erst bei Bedarf erneut aufgebaut und anschliessend wieder abgebaut.

Wie *SMTP* und *HTTP* ist *POP3* ein textbasiertes Protokoll (ASCII) und bietet folgende Befehle:

Befehl	Funktion
`USER xxx`	Wählt das Benutzerkonto auf dem Server aus
`PASS xxx`	Passworteingabe des Benutzers (Klartext)
`STAT`	Zeigt den Status der Mailbox an (Anzahl E-Mails, Gesamtgrösse, etc.)
`LIST (n)`	Listet die Anzahl und Grösse der n-ten E-Mail(s).
`RETR n`	Holt die n-te E-Mail ab
`DELE n`	Löscht die n-te E-Mail.
`NOOP`	Keine Funktion.
`RSET`	`DELE`-Befehle werden zurückgesetzt.
`QUIT`	Beendet die aktuelle Sitzung.
`APOP`	Sichere Anmeldung.
`TOP n x`	Zeigt den Header und die ersten x Zeilen der n-ten Mail an.
`UIDL n`	Zeigt die eindeutige ID der Mail an

Tabelle 9: POP3 Befehle
(Quelle: https://de.wikipedia.org/wiki/Post_Office_Protocol#Kommandos)

4.3.3 IMAP

Bei *IMAP* (***I**nternet **M**essage **A**ccess **P**rotocol*) handelt es sich um ein Netzwerkprotokoll. Wie *POP3* wird es für die Abholung von Mails von einem Mail-Server verwendet.

Der Standard-Port von *IMAP* ist **143/TCP**. Mit einer TLS-abgesicherten Leitung ändert sich der Port standardmässig auf **993/TCP**.

Das Protokoll ist neuer als *POP3* und bietet mehr Funktionen an. Beispielsweise kann mit *IMAP* Einstellungen auf dem Mail-Server gespeichert werden. Zusätzlich lässt sich auch eine Ordnerstruktur erstellen. Im Gegensatz zu POP3 werden die Mails auf den Clients heruntergeladen und bei Änderungen wieder hochgeladen. Da sich alles auf dem Server befindet, funktioniert auch die Synchronisation zwischen mehrere Clients. So behält man auch seine Mails, wenn beispielsweise der Rechner kaputt geht.

Ebenfalls unterstützt *IMAP* sogenannte «Flags». Damit können E-Mails markiert und auf den Clients automatisch sortiert/behandelt werden. So können E-Mails als gelesen, gelöscht, archiviert, etc. markiert werden. Diese Flags werden ebenfalls synchronisiert, so dass auf anderen Clients ebenfalls die Markierungen vorhanden sind.

Und nicht zuletzt verfügt *IMAP* auch über die Funktion, mehrere Sessions gleichzeitig offen zu haben. Hingegen bei POP3 immer nur eine Sitzung offen sein kann. Mit dieser Funktion lassen sich mehrere Konten verbinden und mit denen unabhängig interagieren.

Der Nachteil von IMAP ist, dass es ständig eine Verbindung mit dem Server haben muss. Zudem beansprucht IMAP mehr Ressourcen, vor allem während dem sortieren oder suchen.

Auch IMAP ist ein textbasiertes Protokoll und bietet viele Befehle, unter anderem:

Befehl	**Funktion**
`LOGIN`	Loggt mit `user@domain` und `password` ein.
`LIST`	Listet Mailboxen gemäss Parameter auf.
`STATUS`	Zeigt den aktuellen Status einer Mailbox (Anzahl Nachrichten, ungelesen).
`SELECT`	Wählt eine Mailbox aus.
`SEARCH`	Sucht nach den E-Mails, welcher den Kriterien übereinstimmt.
`FETCH`	Zeigt den Inhalt einer Mail an.
`SET FLAG`	Setzt für die ausgewählten Mails einen Flag (z. B. Gelesen).
`CAPABILIY`	Zeigt alle Möglichkeiten des Servers an.

Tabelle 10: Wichtige IMAP Befehle

5 Sicherheit

5.1 Sicherheitskonzept

Im Sicherheits- bzw. Betriebskonzept wird der Soll-Zustand des Servers definiert. Dabei werden auch vor allem der Datenschutz und die Datensicherheit grossgeschrieben.

Ein Sicherheitskonzept beinhaltet die Analyse von möglichen Gefahren und Risiken und deren Auswirkungen. Das Ziel eines Sicherheitskonzept ist, mögliche Gefahren und Risiken zu eliminieren, oder sich dafür so gut wie möglich zu schützen. Sei es vor böswilligen Angriffen, menschliches/technisches Versagen, oder vor höhere Gewalt.

Ein Sicherheitskonzept beinhaltet folgende vier Punkte:

5.1.1 Schutzobjekte

Es werden die zu schützende Objekte definiert. Zum Beispiel:

Typ	Objekt
Daten	Geheime, vertrauliche, interne oder öffentliche Daten
Systeme	Telefonie, Internetzugang, Server, Datenbanken, etc.
Personen	VRP, CEO, CFO, etc.

Tabelle 11: Mögliche Schutzobjekte

Im Fall von *Fake Inc.* gibt es folgende Schutzobjekte:

Nr.	Schutzobjekt
S1	Geheime/Geschäftskritische Daten
S2	Persönliche Daten
S3	Zahlungsdaten
S4	Server
S5	Datenbank
S6	E-Mail

Tabelle 12: Schutzobjekte von Fake Inc.

5.1.2 Gefahrenanalyse

Eine Gefahr besteht erst, wenn eine Bedrohung und eine Schwachstelle auf ein Schutzobjekt treffen. Mit einer Gefahrenanalyse werden mögliche Gefahren aufgestellt. Dazu kann beispielsweise das BSI-Gefahrenkatalog genutzt werden.

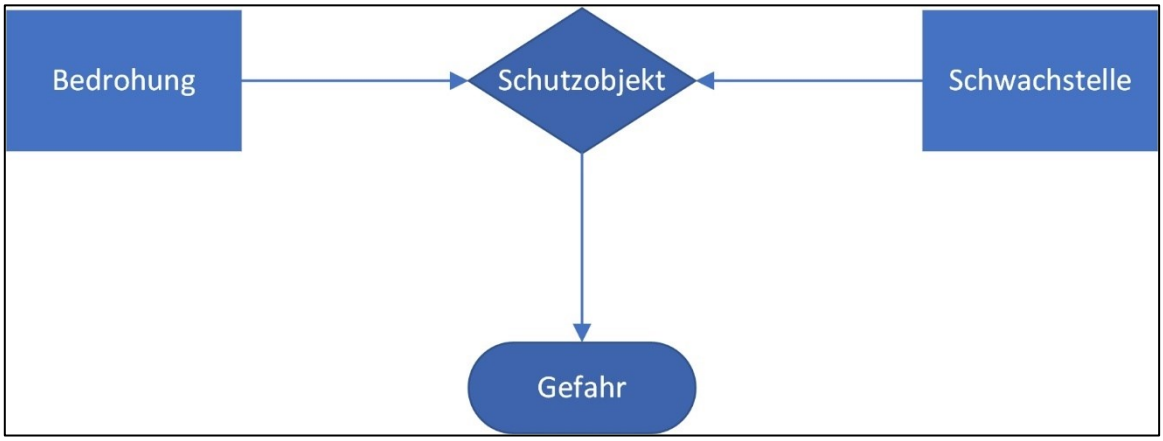

Abbildung 4: Eintritt einer Gefahr

Mögliche Gefahren:

Nr.	Gefahr	Symptome
G1	Virus, Hacker	Verschlüsselte/fehlende Daten, Performance-Einbussen
G2	Veraltetes Betriebssystem	Performance-Einbussen, Bugs, Kompatibilitätsprobleme
G3	Hardwareausfall	(Teil-)Ausfall des Systems
G4	Stromausfall	Kein Strom, nichts läuft
G5	Fehlende Datensicherung	Unabsichtliches Löschen → Nicht wiederherstellbar
G6	Konfigurationsfehler	Kein Zugang (z. B. nur HTTP statt HTTPS), zu viel Zugang

Tabelle 13: Mögliche Gefahren

5.1.3 Risikoanalyse

Das Risiko wird aus der Eintrittswahrscheinlichkeit multipliziert mit dem Schadensausmass berechnet. Mit der Risikoanalyse wird festgestellt wie hoch das Risiko einer Gefahr ist. So können bessere Massnahmen getroffen werden.

Eine Risikoanalyse kann dabei entweder grafisch oder tabellarisch durchgeführt werden:

häufig			G5	
wahrscheinlich				
gelegentlich			G2	G6
entfernt vorstellbar				G1
unwahrscheinlich		G3		
unvorstellbar				G4
↑ Eintritt / Schaden →	unwesentlich	mässig	kritisch	katastrophal

Tabelle 14: Risikoanalyse (grafisch)

Nr.	Gefahr	Wahrsch.	Ausmass	Risiko
G1	Virus, Hacker	3	7	**21**
G2	Veraltetes Betriebssystem	4	5	**20**
G3	Hardwareausfall	2	3	**6**
G4	Stromausfall	1	7	**7**
G5	Fehlende Datensicherung	6	5	**30**
G6	Konfigurationsfehler	4	7	**28**

Tabelle 15: Risikoanalyse (tabellarisch)

Wahrscheinlichkeit	Wertung
unvorstellbar	1
unwahrscheinlich	2
entfernt vorstellbar	3
gelegentlich	4
wahrscheinlich	5
häufig	6

Tabelle 16: Risikoanalyse Legende – Wahrscheinlichkeit

Ausmass	Wertung
unwesentlich	1
mässig	3
kritisch	4
katastrophal	7

Tabelle 17: Risikoanalyse Legende – Ausmass

5.2 Datenschutz

Schutz der persönlichen Daten, wie z. B. Name, Adresse, Geburtsdatum etc. Dabei liegt die Verantwortung bei der jeweiligen Person (Zugriff auf die Daten). Aber die jeweiligen Unternehmen, mit Zugriff auf persönlichen Daten, müssen dafür sorgen, dass diese Daten nicht in falschen Händen geraten.

5.3 Datensicherheit

Damit wird die Sicherheit von Daten (Logfiles, Dokumente, etc.) definiert. Dabei gelten die drei Ziele: Vertraulichkeit, Integrität und Verfügbarkeit. Im Gegensatz zum Datenschutz, muss die Firma für die Datensicherheit sorgen.

Dabei müssen die Server/Daten gemäss den Vorschriften/Gesetzen gesichert werden (EDÖB, DSGVO).

5.4 Datensicherung

Unter Datensicherung (engl. *Backup*) versteht man das Erstellen von Duplikaten der Daten, um mögliche Datenverluste zu vermeiden. Sollte so ein Fall eintreten, kann einfach auf die letzte Sicherung zurückgegriffen werden. Somit erhält man den zuletzt gespeicherten Stand.

Da zwischen Sicherung und Verlust eine gewisse Zeitspanne existiert, führt das zu einem Datenverlust der in dieser Zeit erstellten Daten. Sofern diese nicht anderswo verfügbar sind, sind diese Daten komplett verloren.

Deshalb ist es wichtig, so oft wie möglich ein Backup zu erstellen. Da aber so etwas viel Speicherplatz benötigt (insgesamt doppelte Menge an Daten, evtl. komprimiert), kann nicht jede Stunde eine Sicherung durchgeführt werden. Zudem sollte das System während eines Backups nicht laufen, damit es auch wirklich alles gesichert werden kann.

Es gibt folgende Backup-Typen:

5.4.1 Full Backup

Ein *Full Backup* (zu dt. *Volle Datensicherung*) ist die Sicherung der kompletten Daten. Wird zum ersten Mal ein Backup erstellt, handelt es sich dabei immer um ein Full Backup. Wird nochmals eine Sicherung durchgeführt, werden wieder alle Daten gespeichert, egal ob Daten verändert wurden oder nicht. Diese Art von Backup benötigt den meisten Speicher, kann dafür die Daten zu 100% wiederherstellen (ohne Abhängigkeiten).

5.4.2 Differential Backup

Der *Differentieller Backup* (engl. *Differential Backup*) basiert auf dem Full Backup und sichert somit nur die veränderten Daten seitdem letzten Voll-Backup. Wird nochmals ein differentieller Backup erstellt, wird erneut nur die veränderten Daten seit dem letzten Full Backup gespeichert. Das bedeutet, mit der Zeit speichert die differentielle Datensicherung immer mehr an Daten, sofern kein neue Vollsicherung erstellt wird.

Der Vorteil eines differentiellen Backups ist, dass es nicht allzu viel Speicher verwendet und es lediglich auf den letzten Full Backup basiert.

Der Nachteil hingegen ist ebenfalls der Speicherbedarf, denn je länger kein voller Backup erstellt wird, desto grösser wird die Sicherung.

5.4.3 Incremental Backup

Der letzte Typ ist der *Incremental Backup* (zu dt. *Inkrementelle Datensicherung*). Ähnlich wie zur differentiellen Sicherung, basiert der inkrementelle Backup ebenfalls auf eine vorherige Datensicherung. Aber im Gegensatz zum differentiellen, basiert der inkrementelle immer auf den vorhergehenden Backup. Sprich, der erste inkrementelle Backup basiert auf den Full Backup. Der nächste hingegen, basiert auf die vorherige inkrementelle Sicherung.

Der Vorteil bei dieser Methode ist, dass es am wenigsten Speicher verbraucht, da nur die veränderten Daten seit der letzten Sicherung gespeichert werden.

Der grosse Nachteil ist, um ein Restore durchzuführen, braucht es alle vorherigen inkrementelle Backups bis zur letzten Vollsicherung. Sollte ein Backup fehlen, sind alle Backups nach dem fehlenden nutzlos.

In der folgenden Grafik wird der Unterschied zwischen dem inkrementellen und differentiellen Backup verdeutlicht (beide basieren auf ein wöchentliches Full Backup):

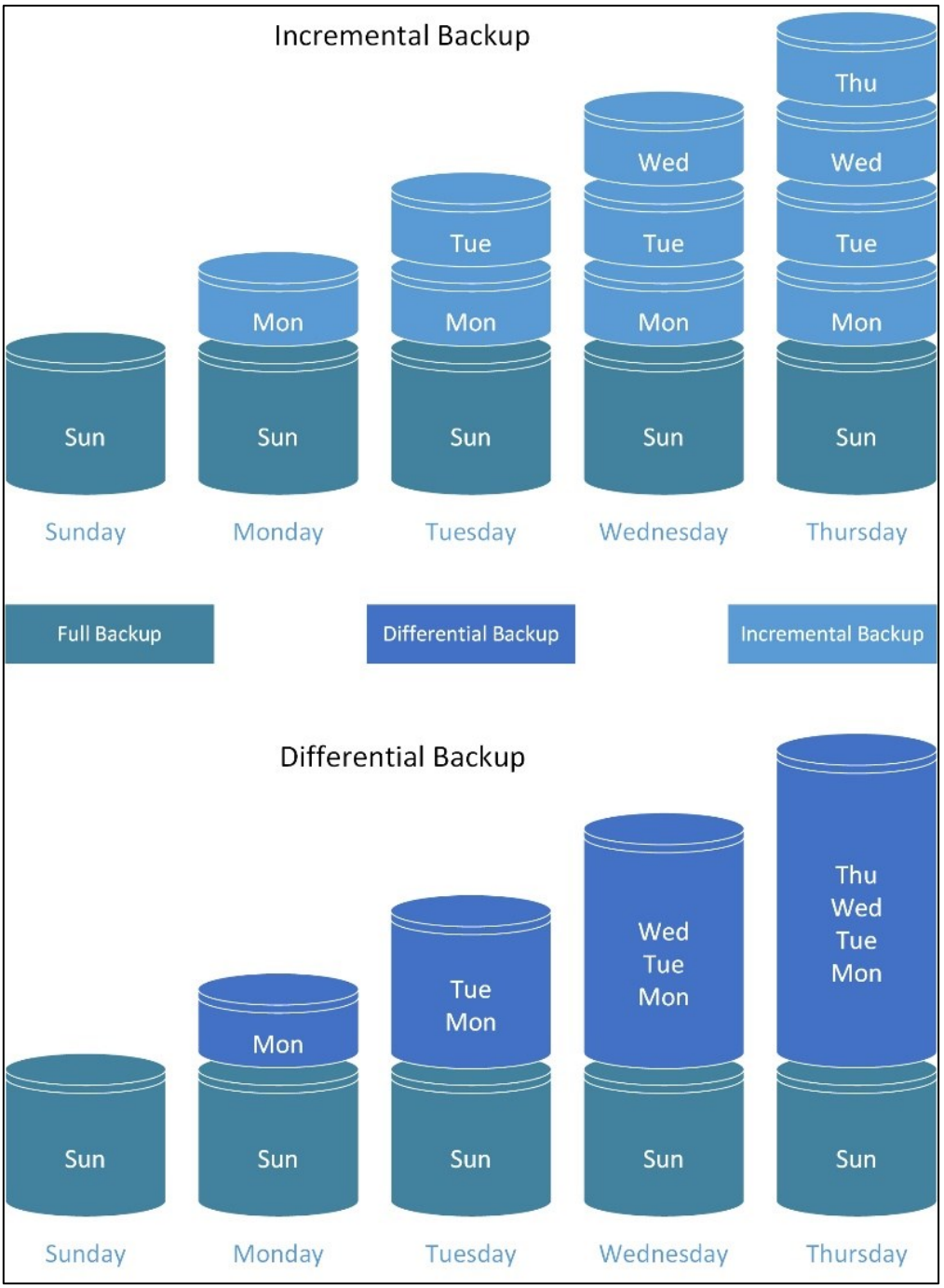

Abbildung 5: Unterschied Incremental/Differential Backup

5.5 Proxy

Ein Proxy wird als Vermittler in einem Netzwerk bzw. zwischen zwei Netzwerken verwendet. Dabei nimmt er auf einer Seite eine Anfrage an und leitet diese entsprechend auf der anderen Seite weiter. Ein Proxy ist sozusagen ein “Man-in-the-Middle”.

Vorteile von Proxys sind unter anderem: Load-Balancing, Verschleierung der IP, Paketfilter, etc.

Es gibt verschiedene Proxy-Typen, welche verschiedene Funktionen ausführen. Zwei oft genutzte Proxy-Typen sind der Dedicated Proxy und der Reverse Proxy.

5.5.1 Dedicated Proxy

Der Dedicated Proxy, teilweise auch Forward Proxy genannt, vermittelt gegen aussen. Dabei sind die wahren IPs der dahinterstehenden Hosts unbekannt. Diese Art von Proxy wird unter anderem im HTTP/S-Bereich als Zwischenspeicher, Zugriffssteuerung, Filterung und SSL-Terminierung verwendet.

5.5.2 Reverse Proxy

Der Reverse Proxy ist das Gegenteil eines Forward Proxys. Dabei kommen die Anfragen von aussen, somit vermittelt der Reverse Proxy nach innen. Hier sind die wahren IPs der Server unbekannt und aussenstehende Geräte sehen nur den Proxy. Diese Art kann unter anderem ebenfalls für die SSL-Terminierung verwendet werden. Es kann aber auch als Load-Balancer oder Paketfilterung verwendet werden.

5.6 Verschlüsselung

Möchte man über das Internet verschlüsselt Daten senden, gibt es grundsätzlich zwei Methoden: Die *Symmetrische Verschlüsselung* und die *Asymmetrische Verschlüsselung.*

5.6.1 Symmetrische Verschlüsselung

Die *Symmetrische Verschlüsselung* stellt die einfachste Art der Kryptographie dar, da es nur ein *Schlüssel*[3] für die Ver- und Entschlüsselung verwendet. Dabei kann der Schlüssel Zahlen, Wörter oder irgendwelche Zeichenketten beinhalten.

Wie bereits erwähnt muss bei diesem Verfahren sowohl für die Ver- und Entschlüsselung derselbe Schlüssel verwendet werden. Somit muss sowohl der Absender als auch der Empfänger den Schlüssel kennen.

Zur Verschlüsselung werden Algorithmen wie AES, RC4, DES, RC5 oder RC6 verwendet. Heutzutage wird hauptsächlich *AES-128*, *AES-192* oder *AES-256* verwendet. Die Zahl am Schluss stellt die Schlüssellänge in Bit dar.

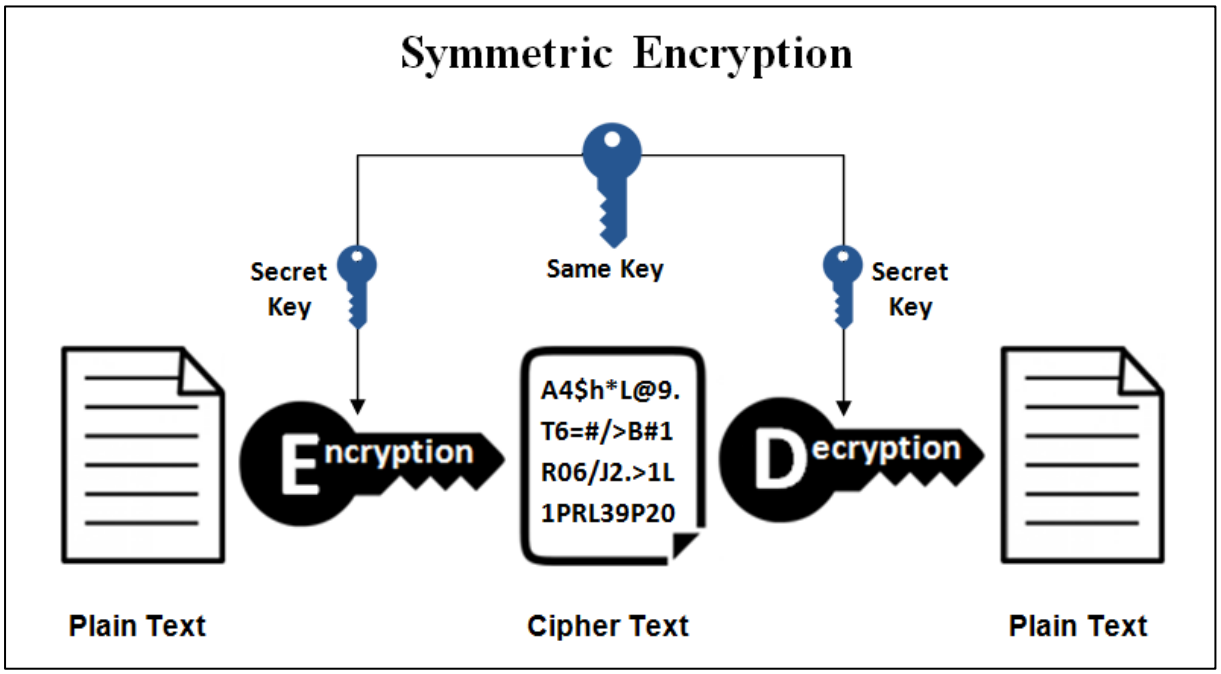

Abbildung 6: Symmetrische Verschlüsselung
(Quelle: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences)

[3] Ein *Schlüssel* ist vergleichbar mit einem Passwort, welches Daten verschlüsselt und entschlüsselt.

5.6.2 Asymmetrische Verschlüsselung

Im Gegensatz zur Symmetrischen Verschlüsselung wird bei der *Asymmetrische Verschlüsselung* zwei Schlüssel, also ein *Schlüsselpaar* verwendet. Ein Schlüsselpaar beinhaltet immer ein *privater Schlüssel* (*private key*) und ein *öffentlicher Schlüssel* (*public key*). Das Schlüsselpaar wird immer zusammen generiert und sind voneinander abhängig.

Der Public Key ist frei zugänglich und wird verwendet, um eine Nachricht oder Datei zu verschlüsseln. Diese Nachricht/Datei kann anschliessend nur vom Private Key Schlüssel entschlüsselt werden. Der Private Key ist geheim zu halten, denn damit lassen sich alle Nachrichten entschlüsseln, welche mit dem dazugehörigen Public Key verschlüsselt wurde.

Umgekehrt funktioniert der Ablauf auch. Wird eine Datei mit dem Private Key verschlüsselt, kann es mit dem Public Key entschlüsselt werde. Dieses Verfahren wird oft genutzt, um seine Identität zu bestätigen, dabei redet man vom *Signieren*.

Asymmetrische Verschlüsselung bietet insgesamt mehr Sicherheit gegenüber symmetrischer Verschlüsselung, da auf ein Schlüsselpaar gesetzt wird. Zudem wird beim Public Key keine Sicherheit benötigt, da dieser für alle zugänglich sein muss.

Diese Methode der Kryptographie wird sehr oft in *Zertifikaten* (siehe 5.7) verwendet. Dabei kommen Algorithmen wie *RSA, DSA, Elliptische-Kurven-Kryptographie* oder *PKCS* zum Einsatz.

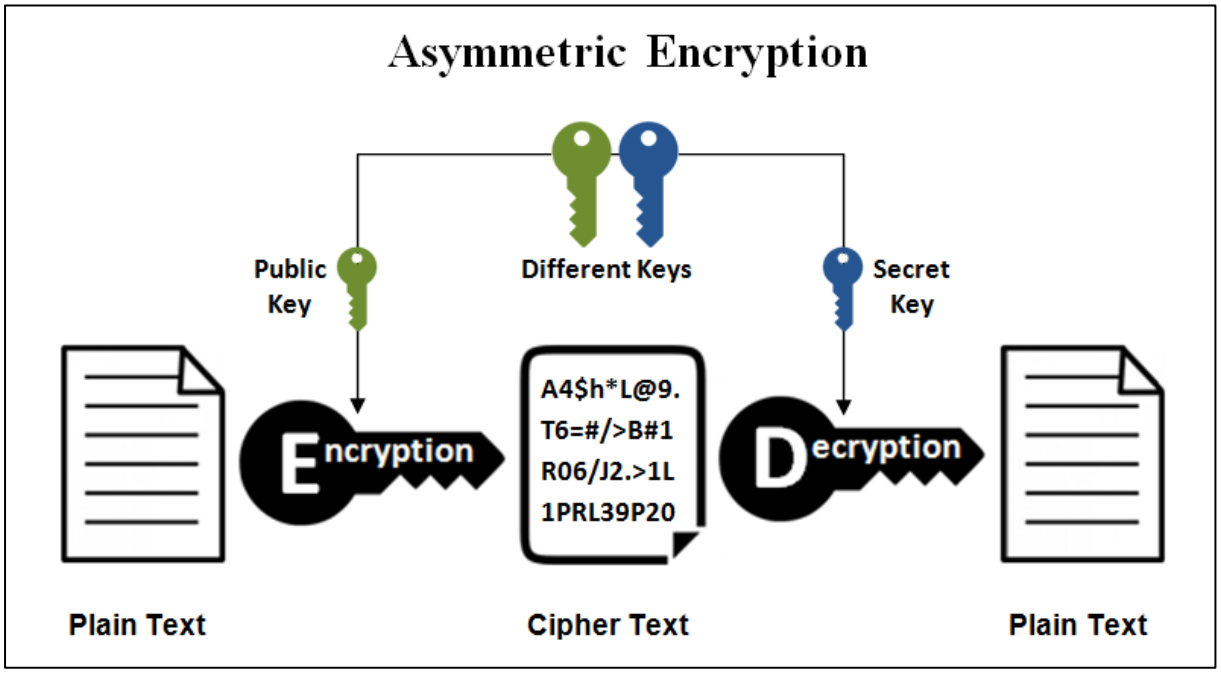

Abbildung 7: Asymmetrische Verschlüsselung
(Quelle: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences)

5.7 Zertifikat

Ein *digitales Zertifikat* erfüllt ähnliche Zwecke wie ein gewöhnliches Zertifikat. Es wird von einer Zertifizierungsstelle ausgestellt und beglaubigt den Besitzer über eine spezifische Funktion. Grundsätzlich dienen sie zur Bestätigung der Authentizität eines öffentlichen Schlüssels.

Man begegnet Zertifikate sehr oft beim Internetsurfen. Sie werden nämlich zur Verschlüsselung der Webseiten benötigt (HTTPS). Sofern eine sichere Verbindung vorhanden ist, kann im Browser das Zertifikat angeschaut werden.

5.7.1 Inhalt

Ein Zertifikat beinhaltet verschiedene Elemente, darunter:

Element	**Beschreibung**
`Aussteller`	Aussteller des Zertifikats (Zertifizierungsstelle)
`Gültig ab`	Beginn der Gültigkeit
`Gültig bis`	Ende der Gültigkeit
`Antragssteller`	Für wen das Zertifikat ausgestellt wurde
`Öffentlicher Schlüssel`	Der dazugehörige Public Key
`Fingerabdruck`	Einzigartiger Fingerabdruck des Zertifikats
`Alt. Antragsstellername`	Alternative Namen/Domains für den Antragssteller

Tabelle 18: Inhalt eines Zertifikats

5.7.2 Arten

Es gibt verschiedene Arten eines Zertifikats. Die drei im Internet am häufigsten anzutreffenden Arten sind:

- «normales» Zertifikat
  - Entweder als Single-/Multi-Domain oder als Wildcard (z. B. `*.example.com`)
- Extended Validation SSL (EV-SSL) Zertifikat
  - Erweiterte Validierung, um das Zertifikat zu erhalten (wird von Unternehmen genutzt)
  - Im Browser wird beim Schloss der Firmenname angezeigt
- Self-signed / Selbst-signiertes Zertifikat
  - Selbst erstelltes Zertifikat (nicht von einem CA), wird/sollte nur intern verwendet werden

5.7.3 Vertrauenswürdigkeit

Zertifikate werden normalerweise von Zertifizierungsstellen (engl. *Certificate Authority, CA*) ausgestellt. Diese wiederum werden von sogenannten *Root-Zertifikaten* erstellt. Am Schluss ergibt das einen Stamm an Zertifikaten, wobei das Root-Zertifikat am Anfang (an der «Wurzel») steht.

Jetzt muss nur noch dem Root-Zertifikat vertraut werden und alle darunterliegenden Zertifikate sind ebenfalls gültig (sofern die Signaturen übereinstimmen).

Die Zertifizierungsstellen arbeiten mit allen Betriebssystemen zusammen. Sowohl Windows, MacOS als auch Linux liefern im Betriebssystem einige *Certificate Authorities* oder *Root-Zertifikaten* mit. Alle Zertifikate welche mit diesen erstellt und signiert wurden, sind somit automatisch gültig und vertrauenswürdig.

5.7.4 Ausstellung

Um ein Zertifikat zu erhalten, muss dieser bei einer vertrauenswürdigen Zertifikatsstelle beantragt werden. Mittlerweile gibt es aber auch kostenlose Zertifikate von einem vertrauenswürdige Zertifikatsstelle, darunter zum Beispiel *Let's Encrypt*. Mit diesem Projekt möchte man für mehr Sicherheit im Internet sorgen, so dass sich jeder ein Zertifikat für seine Webseite holen kann, ohne dabei (viel) Geld auszugeben.

Let's Encrypt wird inzwischen von vielen Providern unterstützt. Zudem ist es auch von verschiedenen Hosting-Applikationen, wie beispielsweise *plesk* oder *cPanel* implementiert, so dass der Benutzer direkt im Webinterface ein Zertifikat erstellen können.

Umso ein kostenloses Zertifikat zu erstellen, müssen lediglich ein paar Informationen über sein Unternehmen eingegeben werden. Anschliessend wird das Zertifikat erstellt und kann direkt verwendet werden. So ein Gratis-Zertifikat hat normalerweise eine Gültigkeit von 90 Tage und wird standardmässig automatisch erneuert.

6 Software und Konfiguration

6.1 Testumgebung

Die Testumgebung ist wie folgt aufgebaut:

Betriebssystem	Ubuntu Server 18.04.02 x64
Arbeitsspeicher	2 GB
Speicher	20 GB (nur für das Betriebssystem und Software)
Hypervisor	VMware Workstation Player 15
Webhosting-Management-Tool	plesk 17.8.11

Tabelle 19: Testumgebung

Als Management Tool wird *plesk* verwendet. Damit lassen sich diverse Domains hosten und verwalten. Dabei kann auf *plesk* praktisch alles erledigt werden, so dass sich nicht mit dem Betriebssystem herumschlagen muss.

Darauf können Software-Updates (für *plesk* und das Betriebssystem), Software-Installationen, etc. durchgeführt werden. Zudem wird *plesk* standardmässig direkt mit MySQL und einem Mail-Server ausgeliefert. Somit können auch diese Services vom Tool aus verwaltet werden.

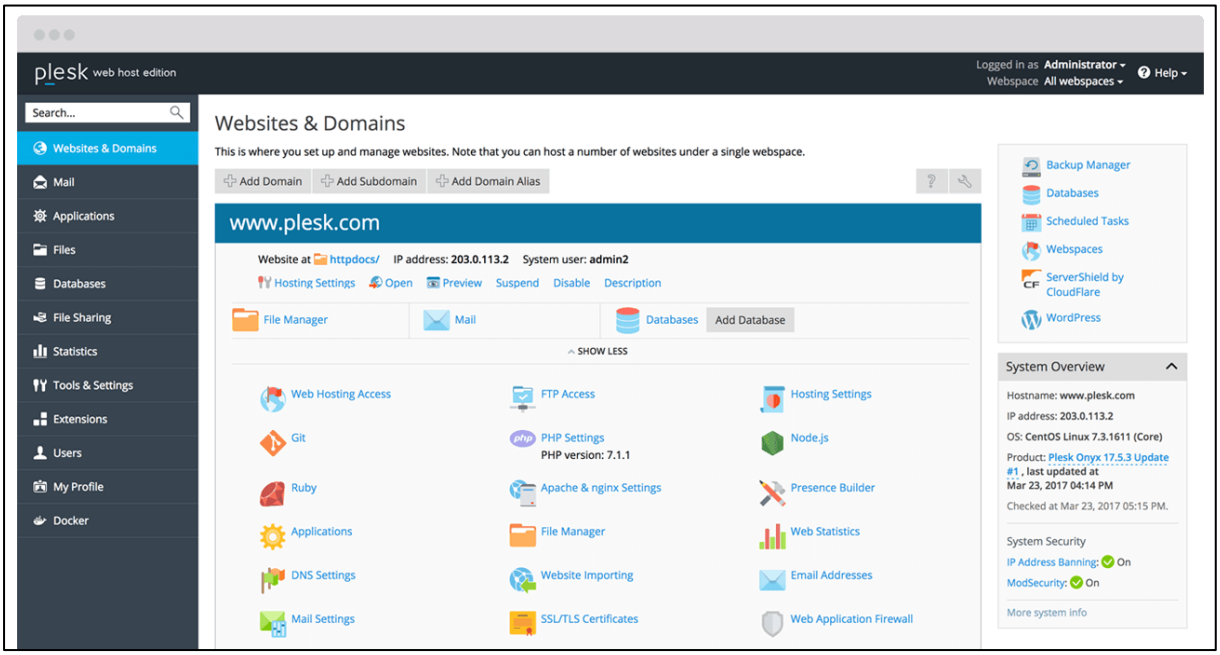

Abbildung 8: Webinterface von plesk
(Quelle: https://www.plesk.com/)

6.2 Web-Server

6.2.1 CMS

Als CMS kommt ***PrestaShop*** zum Einsatz. Es ist ein OpenSource-Projekt speziell für eCommerce ausgelegt.

Nennenswerte Funktionen:

- Einfache Konfiguration von Produkten, Produktpaketen, etc.
- Bestandsführung für jedes Produkt
- Diverse Zahlungsmöglichkeiten einstellbar
- Verschiedene Versandmethoden möglich
- Individuelle Gestaltung des Shops
- Diverse Optionen für rechtliche Informationen

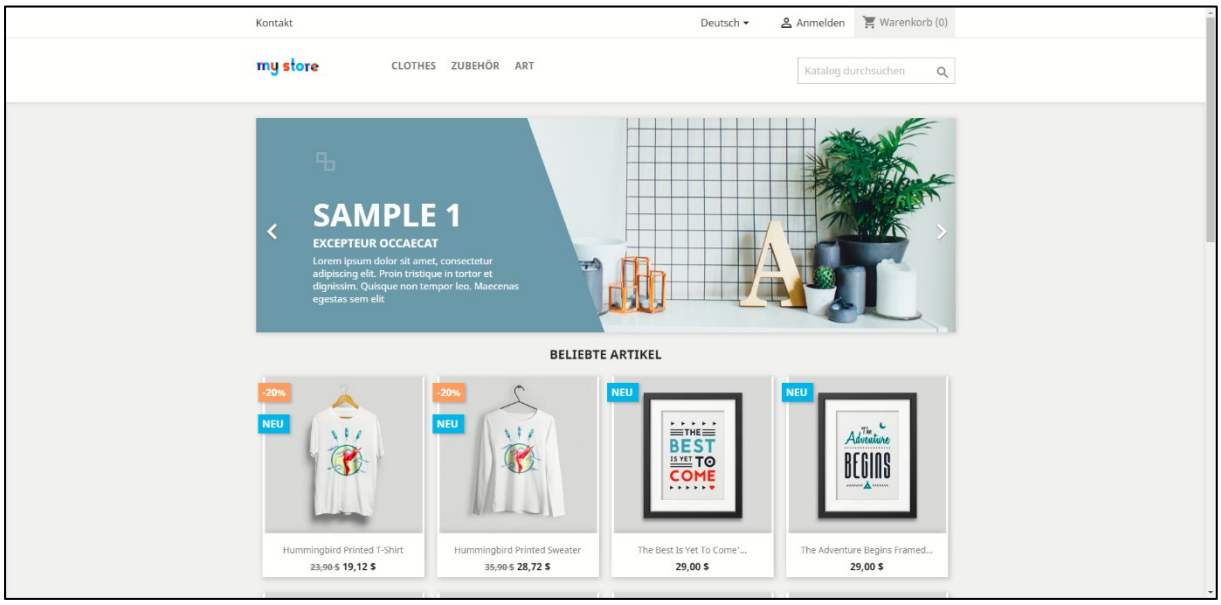

Abbildung 9: Beispiel-Startseite von PrestaShop
(Quelle: http://demo.prestashop.com/de/?view=front)

6.2.2 Virtual Hosting

Mit **Virtual Hosting** wird das gleichzeitige Betreiben mehrerer Domains/IP-Adressen auf einem Server/Host definiert. Dabei gibt es zwei Arten des *virtuellen Hostings*:

In diesem Fall wird das Virtual Hosting von plesk übernommen.

IP-basiertes Virtual Hosting

Beim *IP-basiertes Virtual Hosting* besitzt der Host mehrere Netzwerkschnittstellen. Diese können wiederum mehrere IP-Adresse haben (Interface-Alias). Anschliessend werden die (virtuellen) Server einer IP-Adresse zugeordnet.

Namensbasiertes Virtual Hosting

Bei dieser Variante wird nur eine IP-Adresse für mehrere Domains verwendet. Dabei stützt sich das Verfahren auf das host-Feld im HTTP Header (welches erst ab der Version HTTP/1.1 obligatorisch ist). Dabei findet die Unterscheidung bei der Anwendung statt. Beispielsweise kann ein HTTP Apache Server mehrere (Sub-)Domains haben, welche alle zu verschiedenen Seiten zeigen. Und dass, obwohl es dieselbe IP-Adresse besitzt.

6.3 Datenbank-Server

Der Datenbank-Service wird bereits zusammen mit *plesk* installiert. Dabei handelt es sich standardmässig um ***MySQL***. Bei Bedarf kann auch *PostgreSQL* installiert werden.

Beim Aufsetzen des CMS wird automatisch eine Datenbank und ein Datenbank-Benutzer erstellt. Somit muss dies nicht mehr manuell erstellt werden. Sollte die Nachfrage bestehen, kann logischerweise die Datenbank nachträglich noch angepasst werden.

6.4 Mail-Server

Wie das Web- und Database-Service, wird auch der Mail-Server mit *plesk* vorinstalliert. Sollte man die erweiterte Installation von *plesk* gewählt haben, kann vorher noch entschieden werden, welcher Mail-Service installiert werden soll. Standardmässig verwendet das Management-Tool ***postfix***. Es ist eine freie Software und während der Entwicklung wurden vor allem auf die Sicherheitsaspekte geachtet.

Über das Webinterface von *plesk*, kann schnell und einfach E-Mail-Adressen erstellt werden. Wurden Anti-Spam-Tools wie *SpamAssassin* installiert, können auch zusätzlich Spam-Einstellungen getätigt werden. Je nach Installationsart des Domain-Hosting-Softwares werden bereits Webmails vorinstalliert. *Plesk* unterstützt derzeit zwei Webmails: *Roundcube* und *Horde*.

6.5 Anbindungen

Je nach Hosting differenziert sich die Anbindung des Web-Servers. In Fall, dass der Shop bei einem Provider gehostet wird, ist der Web-Server nur vom Internet aus zugänglich. Zudem sollte der Web-Server mit einem Zertifikat gesichert werden und nur über HTTPS (Port 443) zugänglich sein.

Der Web-Server resp. das CMS (*PrestaShop*) benötigt eine Datenbank, um richtig zu funktionieren. In jedem Fall **sollte** die Datenbank nur lokal zugänglich sein. Da alle Services auf demselben Server sind, sollte somit keine Verbindung der Datenbank nach aussen führen, nicht einmal ins lokale Netzwerk. Das CMS benötigt eine MySQL-Datenbank, der Standard-Port ist 3306.

Beim Mail-Server sieht auch wieder anders aus. Der Server muss mit dem Internet kommunizieren können, ansonsten kann nur intern E-Mails versendet werden. Ausserdem erfüllt dieser Service zwei Funktionen. Es dient sowohl als SMTP- als auch als IMAP-Server. Zu den Clients wird mit IMAP (143 / 993) kommuniziert. Nach aussen benötigt es SMTP (25 / 465). Auch hier sollte optimalerweise ein Zertifikat verwendet werden.

6.6 Protokollierung

Grundbegriffe:

Protokollierungsgrad
Viele *Daemons* (Linux-Services / -Applikationen) können verschieden «stark» loggen. Je nach Einstellung wird mehr oder weniger geloggt. Grundsätzlich gibt es drei bis vier Stufen:

- Fehler → Nur Fehler werden geloggt
- Warnungen → Nur Fehler und Warnungen werden geloggt
- Infos → Allgemeine Infos, Warnungen und Fehler werden geloggt
- Verbose → Alle Aktivitäten werden geloggt

Die Protokollierung spielt eine sehr wichtige Rolle in Sachen Security. Denn Logs geben schnell Aufschluss über ein System, je nach Protokollierungsgrad. Ganz wichtig werden sie, sobald etwas nicht mehr richtig funktioniert, oder jemand/etwas in das System eingedrungen ist. Beim Letzterem können Logs sogar im Fall einer Gerichtsverhandlung als Beweismittel verwendet werden. Deshalb ist es wichtig, dass die Logs richtig konfiguriert sind und optimalerweise zentral abgelegt und gesichert werden.

Je nach Grad, stehen in den Logs alle Aktivitäten eines Service. Beispielsweise bei einem SSH-Service werden alle Verbindungen mit den entsprechenden IP-Adressen geloggt. Neben solchen Infos, werden auch Fehler und Warnungen geloggt. So kann der Administrator entsprechend die Fehler beseitigen, zum Beispiel bei einer Fehlkonfiguration.

Je nach Betriebssystem und Applikation werden Logs anders eingestellt, geschrieben und gespeichert. In Windows gibt es ein zentrales Tool namens *Ereignisanzeige*. Diesen zu verwenden wird den Applikationen zwar nicht vorausgesetzt, aber es ist ziemlich praktisch dieses Tool zu nutzen. Denn so befinden sich alle Logs an einem Ort und der Admin/User kann sie besser finden.

Auf Linux sieht das Ganze schon ein wenig anders aus. Da Linux hauptsächlich über das Terminal verwendet wird (Server-Versionen), loggen so ziemlich alle Services/*Daemons*. Dabei gibt es ebenfalls ein zentraler Speicherort: `/var/log`.

Möchte man beispielsweise den Speicherort der Logdateien ändern, muss das für jede Software einzeln erledigt werden. Denn jede Applikation hat seine eigene Log-Einstellungen. Auch der Protokollierungsgrad muss für jede Software manuell eingestellt werden.

6.7 Zugriffsberechtigung

6.7.1 Gruppen

	Gast	Kunde	Verkäufer	Admin	IT
Gäste	x				
Kunden (Login)		x			
Verkäufer 1–12			x		
CEO				x	
CFO				x	
HR				x	
IT-Support					x

Tabelle 20: Sicherheitsgruppen

6.7.2 Berechtigungen

Die Berechtigungen sind ziemlich simpel gehalten. Für mehr Sicherheit können die Berechtigungen entsprechend auf einer tieferen Stufe erfolgen.

In diesem Beispiel sind es fünf Berechtigungsgruppen: Kundenbereich, Backoffice, Zahlungen, Personal und IT-System.

Der Kundenbereich darf nur vom Kunden und höchstens noch von einem IT-Supporter eingesehen werden, ansonsten kann es die Privatsphäre (Datenschutz) des Kunden verletzt werden. Das Backoffice ist das Backend des CMS, dort werden alle Einstellungen für den Webshop getroffen, darunter die Produkte, etc. hinzugefügt und angepasst. Für diesen Bereich hat nur die Administration und der IT-Support Zugriff. Die Verkäufer können höchstens auf die Zahlungen zugreifen. Die Berechtigung für personelle Daten und das IT-System sind selbsterklärend.

	Kundenbereich	Backoffice	Zahlungen	Personal	IT-System
Gast					
Kunde	x				
Verkäufer			x		
Admin		x	x	x	
IT	x	x	x	x	x

Tabelle 21: Berechtigungsmatrix

7 Tests

Um die Zuverlässigkeit des Online-Shops zu testen, müssen verschiedene Tests durchgeführt werden. Dabei werden sie in vier verschiedene Kategorien unterschieden:

- Technischer Test
- Applikationstest
- Sicherheitstest
- Lasttest

Mögliche Status:

- **OK** → OK
- **NOK** → Not OK

7.1 Technisch

Ein *Technischer Test* ist dafür ausgelegt die Infrastruktur und Hardware zu testen. Da es bei *Fake Inc.* um ein Hosted Server geht, muss lediglich das Routing getestet werden. Tests über die Stromversorgung, Belüftung, Burn-In-Tests, etc. können nicht von extern durchgeführt werden. Somit müssen diese Punkte vom Provider getestet werden.

7.1.1 Routing

ID / Bezeichnung / Status	`TR-001`	`Erreichbarkeit Server`	`-`
Beschreibung	Der Server kann vom Internet aus erreicht werden		
Testvoraussetzung	Server ist online		
Testschritte	1. Befehlszeile öffnen 2. Befehl eingeben: `ping [Server-IP]`		
Erwartetes Ergebnis	Server kann erreicht werden / Antwort kommt zurück		
Effektives Ergebnis			
Fehler (falls NOK)			

7.2 Applikation

Be einem *Applikationstest* wird effektiv die Funktionalität der Serverapplikationen überprüft. Dabei wird geachtet, ob die Software richtig konfiguriert wurde, die Zugriffe vorhanden sind, etc.

7.2.1 Web-Server

ID / Bezeichnung / Status	`AW-001`	`Erreichbarkeit Websites`	`-`
Beschreibung	Alle Webseiten können angezeigt/erreicht werden		
Testvoraussetzung	Server ist online, Webserver eingerichtet		
Testschritte	Webseiten öffnen		
Erwartetes Ergebnis	Können ohne (HTTP-)Fehler geöffnet werden (Code `200 OK`)		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	`AW-002`	`Zugriff Kundenportal`	`-`
Beschreibung	Das Kundenprotal kann geöffnet werden, sofern man als Kunde eingeloggt ist.		
Testvoraussetzung	Server ist online, Online-Shop eingerichtet		
Testschritte	1. Online-Shop aufrufen 2. Als Kunde einloggen / nicht einloggen 3. Kundenportal öffnen		
Erwartetes Ergebnis	• Portal kann ohne Fehler geöffnet werden • Nur eingeloggte User können den Kundenbereich öffnen		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	`AW-003`	`Zugriff Verkaufsportal`	-
Beschreibung	Der Verkaufsprotal kann nur von eingeloggtem Personal zugegriffen werden.		
Testvoraussetzung	Server ist online, Online-Shop eingerichtet		
Testschritte	1. Online-Shop öffnen 2. Als Kunde/Personal einloggen 3. Verkaufsportal öffnen		
Erwartetes Ergebnis	• Portal kann ohne Probleme geöffnet werden • Nur Personal hat Zugriff		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	`AW-004`	`Verschlüsselte Verbindung I`	-
Beschreibung	Die Website besitzt ein gültiges Zertifikat		
Testvoraussetzung	Server ist online, Online-Shop eingerichtet, Zertifikat vorhanden		
Testschritte	1. Online-Shop aufrufen 2. Zertifikat im Browser überprüfen		
Erwartetes Ergebnis	Gültiges Zertifikat vorhanden		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	`AW-005`	`Verschlüsselte Verbindung II`	-
Beschreibung	Die Website kann mit HTTPS aufgerufen werden		
Testvoraussetzung	Server online, Shop eingerichtet, Zertifikat eingerichtet		
Testschritte	1. Online-Shop aufrufen 2. Verbindung im Browser überprüfen		
Erwartetes Ergebnis	Webseite funktioniert (Status `200 OK`) Verbindung ist verschlüsselt		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	AW-006	Verschlüsselte Verbindung III	-
Beschreibung	HTTP wird zu HTTPS weitergeleitet		
Testvoraussetzung	Server online, Shop eingerichtet, Zertifikat eingerichtet		
Testschritte	3. Online-Shop mit HTTP/80 aufrufen 4. Verbindung überprüfen		
Erwartetes Ergebnis	Weiterleitung zu HTTPS		
Effektives Ergebnis			
Fehler (falls NOK)			

7.2.2 Mail-Server

ID / Bezeichnung / Status	AM-001	Login E-Mail / Webmail	-
Beschreibung	Login im Webmail mit der E-Mail-Adresse		
Testvoraussetzung	Server online, Mail-Server konfiguriert, E-Mail-Adresse		
Testschritte	1. Webmail öffnen 2. Login mit E-Mail-Adresse und Passwort		
Erwartetes Ergebnis	Login funktioniert		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	AM-002	Mails versenden	-
Beschreibung	Mails können von der E-Mail-Adresse versendet werden		
Testvoraussetzung	Server online, Mail-Server konfiguriert, E-Mail-Adresse		
Testschritte	1. Webmail öffnen 2. Login mit E-Mail-Adresse und Passwort 3. Mail versenden		
Erwartetes Ergebnis	E-Mail wird versendet, Empfänger empfängt		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	AM-003	Mails empfangen	-
Beschreibung	Mails können von der Mailbox empfangen werden		
Testvoraussetzung	Server online, Mail-Server konfiguriert, E-Mail-Adresse		
Testschritte	1. Webmail öffnen 2. Login mit E-Mail-Adresse und Passwort		
Erwartetes Ergebnis	Mails sollten automatisch heruntergeladen werden		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	AM-004	Spam Mails	-
Beschreibung	Spam Mails werden erkannt und entsprechend gehandhabt		
Testvoraussetzung	Server online, Mail-Server konfiguriert, E-Mail-Adresse		
Testschritte	1. Webmail öffnen 2. Login mit E-Mail-Adresse und Passwort		
Erwartetes Ergebnis	Einkommende Spam-Mails werden abgefangen		
Effektives Ergebnis			
Fehler (falls NOK)			

7.2.3 Datenbank-Server

ID / Bezeichnung / Status	AD-001	Erreichbarkeit DB-Server	-
Beschreibung	DB-Server ist online und kann erreicht werden		
Testvoraussetzung	Server online, DB-Server eingerichtet		
Testschritte	1. phpMyAdmin oder Befehlszeile öffnen 2. Einloggen / mit DB-Server verbinden		
Erwartetes Ergebnis	DB-Server ist erreichbar, Login funktioniert		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	AD-002	Daten können abgefragt werden	-
Beschreibung	DB-Server ist online und kann erreicht werden		
Testvoraussetzung	Server online, DB-Server eingerichtet		
Testschritte	1. phpMyAdmin oder Befehlszeile öffnen 2. Einloggen / mit DB-Server verbinden 3. SELECT-Abfrage durchführen		
Erwartetes Ergebnis	Abfrage funktioniert		
Effektives Ergebnis			
Fehler (falls NOK)			

7.3 Sicherheit

Bei *Sicherheitstest* wird die Sicherheit des Servers überprüft. Dabei wird auf die Verfügbarkeit, Vertraulichkeit und auf die Integrität geachtet.

7.3.1 Verfügbarkeit

ID / Bezeichnung / Status	`SA-001`	`24/7 Verfügbarkeit`	`-`
Beschreibung	Der Server ist 24/7 verfügbar.		
Testvoraussetzung	Server und alle Services konfiguriert und online		
Testschritte	Server und Services starten Kontrolle nach x Tagen (z. B. 1 Woche)		
Erwartetes Ergebnis	Server und Services laufen durchgehend, keine/kaum Ausfälle		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	`SA-002`	`Verhalten unter Last`	`-`
Beschreibung	Der Server soll auch unter Last normal laufen		
Testvoraussetzung	Server und alle Services konfiguriert und online		
Testschritte	1. Server und Services starten 2. Traffic generieren (durch Testing-Software) 3. Kontrolle Funktionalität		
Erwartetes Ergebnis	Server und Services laufen, keine langen Wartezeiten.		
Effektives Ergebnis			
Fehler (falls NOK)			

7.3.2 Vertraulichkeit

ID / Bezeichnung / Status	SC-001	Geschützte Daten I	-
Beschreibung	Schützenswerte Daten sind sicher vor fremden Zugriff.		
Testvoraussetzung	Server und Services konfiguriert und online		
Testschritte	Auf schützenswerte Daten zugreifen (als Gast)		
Erwartetes Ergebnis	Kein Zugriff		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	SC-002	Geschützte Daten II	-
Beschreibung	Schützenwerte Daten können vom Personal abgerufen werden		
Testvoraussetzung	Server und Services konfiguriert und online		
Testschritte	1. Website aufrufen 2. Als Personal einloggen 3. Auf schützenswerte Daten zugreifen		
Erwartetes Ergebnis	Zugriff gewährt		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	SC-003	Geschützte Daten III	-
Beschreibung	Berechtigungen für bestimmte geschützte Daten		
Testvoraussetzung	Server und Services konfiguriert und online		
Testschritte	1. Auf Website einloggen 2. Als User einloggen (Kunde, Verkäufer, HR) 3. Auf geschützte Daten zugreifen		
Erwartetes Ergebnis	Zugriff nur für Verkäufer resp. nur für HR		
Effektives Ergebnis			
Fehler (falls NOK)			

7.3.3 Integrität

ID / Bezeichnung / Status	SI-001	Zertifikat Datum	-
Beschreibung	Das Zertifikat aktuell und gültig.		
Testvoraussetzung	Server und Services konfiguriert/online, Zertifikat vorhanden		
Testschritte	Zertifikat überprüfen		
Erwartetes Ergebnis	Zertifikat ist aktuell und gültig		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	SI-002	Zertifikat Domain	-
Beschreibung	Die Domain im Zertifikat stimmt mit der Website überein		
Testvoraussetzung	Server und Services konfiguriert/online, Zertifikat vorhanden		
Testschritte	Zertifikat überprüfen		
Erwartetes Ergebnis	Domain im Zertifikat stimmt mit der URL überein		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	SI-003	DNS (sofern vorhanden)	-
Beschreibung	Nur benötigte und aktuelle DNS-Einträge vorhanden		
Testvoraussetzung	DNS-Server konfiguriert und online		
Testschritte	DNS-Einträge in der Konfiguration überprüfen		
Erwartetes Ergebnis	DNS-Records korrekt, benötigt und aktuell		
Effektives Ergebnis			
Fehler (falls NOK)			

7.4 Lasttest

ID / Bezeichnung / Status	LT-001	Lasttest I	-
Beschreibung	Server bleibt unter voller Last verfügbar		
Testvoraussetzung	Server und Services konfiguriert und online		
Testschritte	Viele Benutzer nutzen Online-Shop gleichzeitig		
Erwartetes Ergebnis	Server bleibt stabil, evtl. ein bisschen längere Ladezeiten		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	LT-004	Lasttest II	-
Beschreibung	Server bleibt unter voller Last verfügbar		
Testvoraussetzung	Server und Services konfiguriert und online		
Testschritte	Web Capacity Analysis Tool verwenden		
Erwartetes Ergebnis	Server bleibt stabil, evtl. ein bisschen längere Ladezeiten		
Effektives Ergebnis			
Fehler (falls NOK)			

ID / Bezeichnung / Status	LT-003	Lasttest III	-
Beschreibung	Server bleibt unter voller Last verfügbar		
Testvoraussetzung	Server und Services konfiguriert und online		
Testschritte	Lasttest as a Service nutzen		
Erwartetes Ergebnis	Server bleibt stabil, evtl. ein bisschen längere Ladezeiten		
Effektives Ergebnis			
Fehler (falls NOK)			

8 Reflexion

8.1 Lernprozess

Im Modul 239 habe ich sowohl bekannte als auch neue Sachen angetroffen. Beispielsweise waren mir die Themen wie *DNS*, *Sicherheitskonzept*, *Verschlüsselung* und *Zertifikat* bereits bekannt, aufgrund meiner Arbeit im Betrieb, aber auch durch vorherige Module.

Neu waren mir allerdings die Themen HTTP/S, SMTP, IMAP, POP3, etc. Nicht, dass ich davor noch nie etwas davon gehört habe, aber ich hatte sie vorher noch nie vertieft behandelt. Mir war zwar klar was eine URL ist, aber ich hatte es noch nie vertieft angeschaut. Oder die verschiedenen HTTP-Methoden wie GET und POST, welche beide für die Ressourcen-Abfrage genutzt werden kann. Ebenfalls welches ich nie vertieft angeschaut habe, sind Cookies und wie sie genau übermittelt werden.

Neben dem technischen Stoff habe ich einiges organisatorisches gelernt. Wie zum Beispiel wie eine wissenschaftliche Arbeit aufgebaut ist, respektiv was es für Standards und «Best Practises» gibt. Zum Beispiel ist die Nummerierung von Überschriften im DIN 8005 genormt. Dieser besagt beispielsweise, dass die Nummerierung keinen Punkt am Schluss haben darf. Ausser für die oberste Überschrift kann ein Punkt angefügt werden, muss aber nicht. Ebenfalls gibt es eine Faustregel für Schriftgrössen, damit die Schrift nicht allzu gross oder klein ist. Solche Sachen und noch vieles mehr habe ich speziell für diese Dokumentation nachgeschlagen, da sie doch ein wenig gross wurde.

Nicht zu vergessen sind die Tools, die mir dabei geholfen haben. Anfangs haben wir mit Google Docs gearbeitet, da unsere Lehrperson die Lernumgebung auf Google Drive aufgebaut hat. Aber ich habe ziemlich schnell gemerkt, dass Google Docs im Vergleich zu Microsoft Word viele Funktionen fehlen, wie zum Beispiel automatisch das aktuelle Datum einfügen. Oder die Absatzmarken anzuzeigen.

8.2 Positive Punkte

Meiner Meinung nach ist mir die Dokumentation ziemlich gelungen. Anfangs hatte ich mit weniger Seiten gerechnet, bis ich ungefähr bei der Hälfte bereits 30 Seiten hatte.

Ausserdem waren die Themen, welche ich bereits kannte, ziemlich einfach zu behandeln. Aber auch der neue Stoff war nicht gerade allzu schwierig zu bearbeiten. Das aus dem Grund, weil ich ein Teil schon kannte, oder weil das Thema nicht gerade schwierig war.

8.3 Negative Punkte

Wie ich bereits kurz im Lernprozess festgehalten habe, war Google Docs das grösste Problem. Ich habe die Dokumentation ungefähr bis zur Hälfte in Google Docs geschrieben, bevor ich alles zu Microsoft Word gezügelt habe (→ ab ca. 20 Seiten fing Google Docs an zu ruckeln).

An sich hat Google Docs schon seine Daseinsberechtigung. Und für gewisse Sachen ist Google Docs schon praktisch, aber für eine Dokumentation ist es meiner Meinung nach eher unpraktisch, da viele Funktionen fehlen, welches viel Zeit sparen würde.

Unter anderem kann Google Docs keine Absatzmarker anzeigen, was meiner Meinung nach essenziell ist. Ich schreibe Dokumente immer mit Absatzmarker, da man so alle Zeichen sieht, auch die Leerzeichen und -zeilen. Zudem kann Docs kein automatisches Datum einfügen, man muss es jedes Mal bearbeiten. Auch die Nummerierung in der Überschrift muss manuell gemacht werden, da es dafür keine Funktion gibt.

Ich weiss, dass es zwar Addons, Scripts, etc. für Google gibt, aber sie erfüllen meist nicht die vollen Anforderungen. Ausser man scriptet es selbst, was wiederum wieder Zeit kostet.

An sich kein «Flop», aber aus meiner Sicht ist der Kompetenzraster suboptimal. Die Kompetenzen waren für mich eher undeutlich formuliert, weswegen ich auch nie genau wusste was verlangt wurde. Zwar erhielten wir eine Hilfestellung für die einzelnen Kompetenzen, aber teilweise habe die auch nicht gerade weitergeholfen, oder waren schlichtweg zu aufwendig.

Ansonsten hatte ich keine Schwierigkeiten während des Moduls. Höchstens wäre noch die mangelnde Zeit erwähnenswert.

8.4 Verbesserungen

Zum Schluss noch einige Verbesserungsmöglichkeiten und -vorschläge:

Vor allem aufgrund der mangelnden Zeit konnte ich einige Themen nicht genug vertiefen, geschweige denn verständlich(er) beschreiben. Zudem kommt noch dazu, dass ich während dem Unterricht nicht so konzentriert arbeiten kann, wie im Büro oder zuhause. Das liegt einfach daran, dass es entweder zu laut/störend ist, oder weil wir immer wieder wegen Theorie-Inputs gestört wurden – nicht, dass die Inputs schlecht waren oder so, nur wurde man immer wieder aus dem Fluss rausgezogen, so dass man sich wieder zuerst einfinden musste.

Es wird bestimmt noch andere Sachen geben, die ich im Nachhinein besser hätte machen können, wie zum Beispiel die Strukturierung der Dokumentation. Am Anfang hatte ich die Dokumentation nach den Kompetenzen gegliedert, um es der Lehrperson einfacher zu machen. Weil im letzten Modul (M300: Plattformübergreifende Dienste) erhielt man Abzug, wenn man es nicht nach Kompetenzen, sondern nach Themen strukturiert hatte. Aber in diesem Modul wiederum, meinte die Lehrperson es wäre schlauer nach Themen zu ordnen. Also habe ich die gesamte Dokumentation umstrukturiert. Das passierte genau zu diesem Zeitpunkt, bei dem ich zu Word zügelte, somit war es nicht allzu mühsam alles zu umsortieren.

Nun zu den Sachen, was mich am Modul störte und wie man es verbessern könnte. Ein Punkt welches ich umständlich fand, war dass wir auf verschiedenen Plattformen arbeiten mussten. Ein Teil war auf dem BSCW und ein Teil auf Google Drive. Bei mir kam dann noch Word hinzu, welches auf meinem OneDrive lag. Meiner Meinung nach, wenn wir schon von der Schule die Mittel kriegen, könnte man auch einfach ein SharePoint verwenden. Es erzielt den gleichen Zweck und jeder hat es bereits. Zudem nutzen viele Firmen Microsoft Office und SharePoint. Somit arbeitete man mit Mitteln, welche unter Umständen auch im Betrieb verwendet wurde.

Ein anderer grosser Punkt war der Kompetenzraster. Aus meiner Sicht könnte man den Kompetenzraster vergessen und stattdessen ein anderes Bewertungssystem verwenden. Im vorherigen Modul erhielten wir ein Bewertungsbogen, (angeblich) ähnlich zur IPA. Diese beinhalteten auch viel spezifischere Punkte, welche man abarbeiten musste. So war klar was getan werden musste. Anders kann es auch mit einem einfachen Anforderungsblatt gelöst werden. Darauf steht Punkt für Punkt welche Themen zu bearbeiten sind und was die Minimalanforderung ist.

Logischerweise war nicht alles schlecht, im Gegenteil, es sind nur die wenigen Punkte. Im Grossen und Ganzen fand ich das Modul sehr spannend und lehrreich.