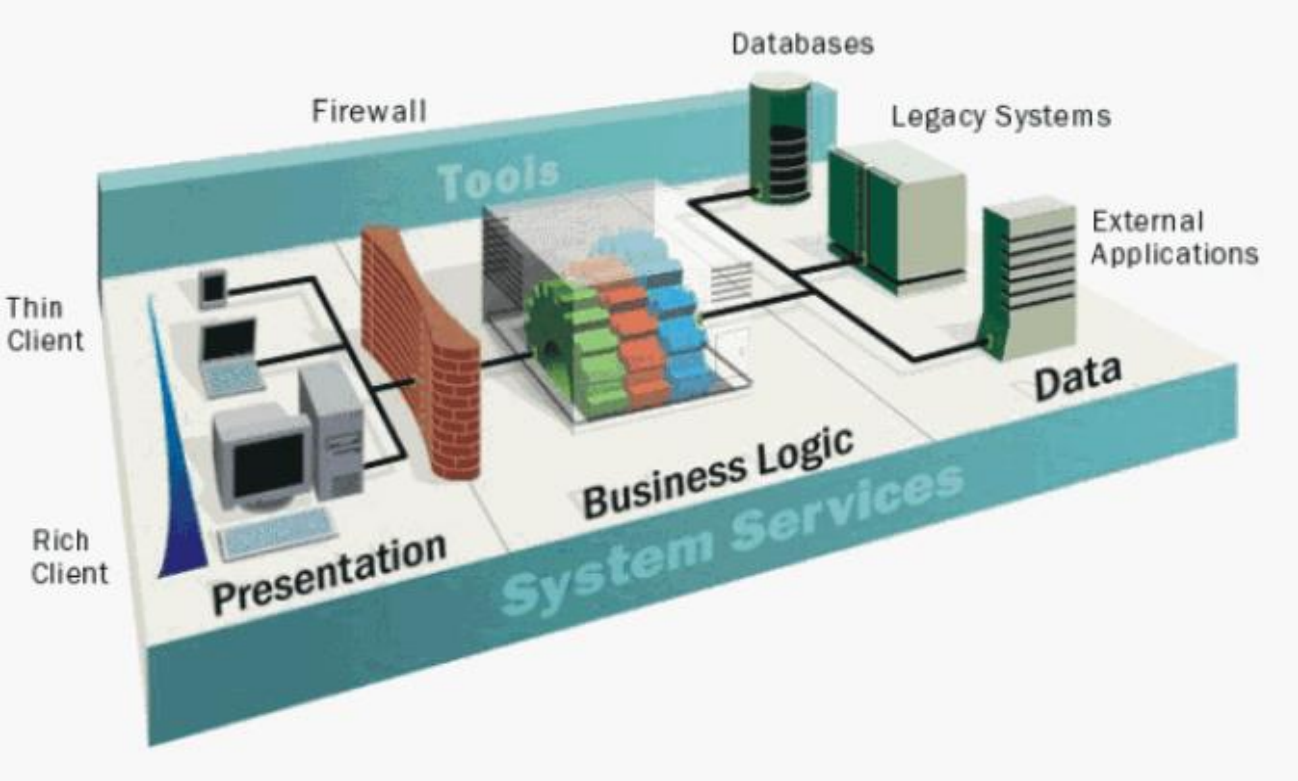
M133

Web-Applikation gemäss Vorgabe mit einer Programmiersprache realisieren und testen

IT-Architekturen



Presentation Layer (PL)

- beinhaltet das Mensch- Maschinen Interface (MMI oder GUI).
- Stellt die von der Business Layer aufbereiteten Daten dar
- Eingaben werden grundsätzlich im Presentation Layer getätigt.
- Presentation Layer beinhaltet nur Code für die Navigation und allenfalls die Überprüfung von User Eingaben

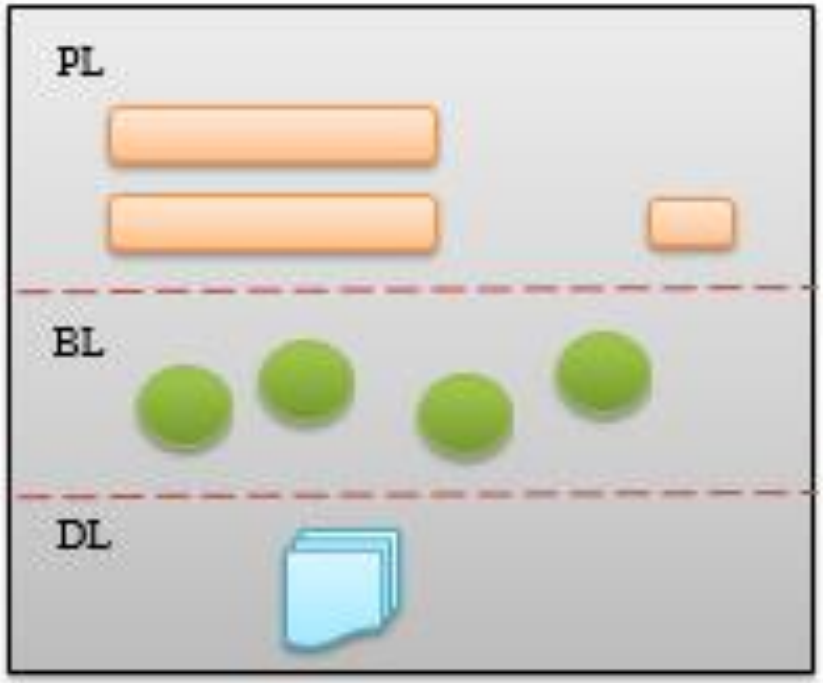
Business Layer (BL)

- Beinhaltet den eigentlichen Programm Code einer Applikation

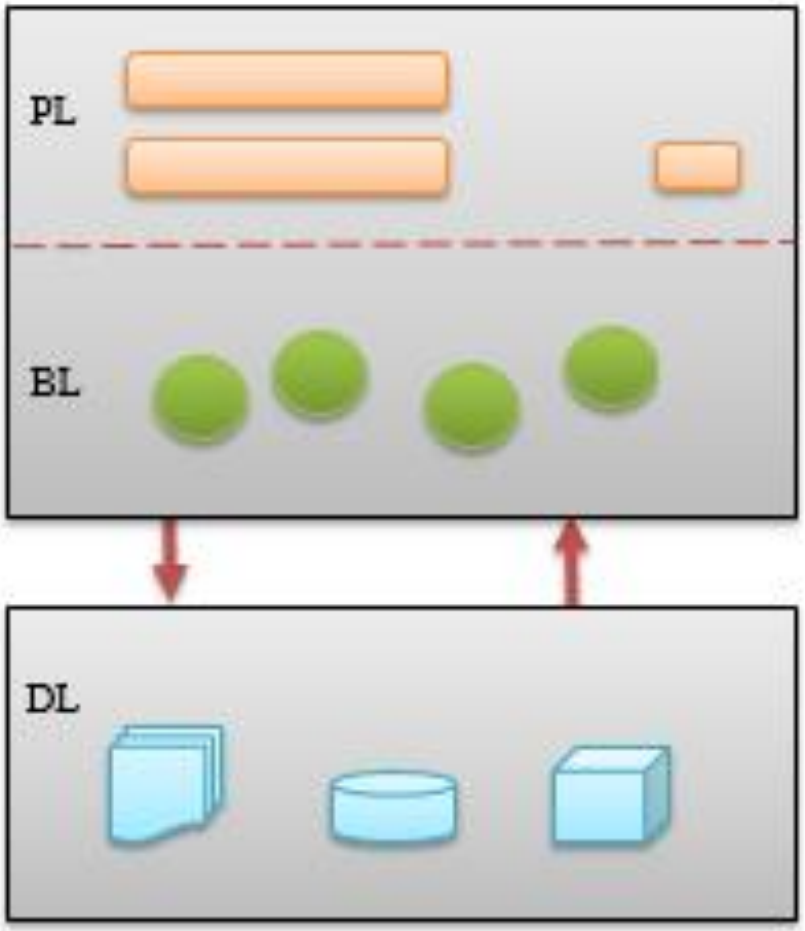
Data Layer (DL)

- Verwaltet die Daten der Applikation
- Beinhaltet Filesystem, Datenbanken, etc.

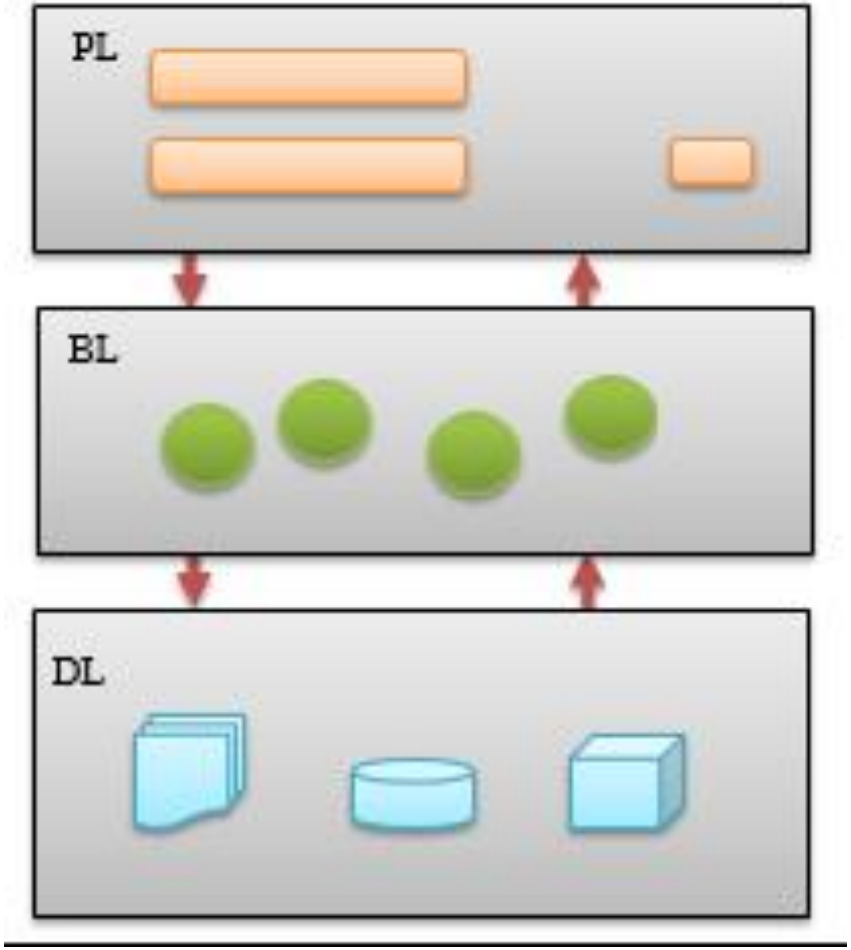
1 Tier Architektur



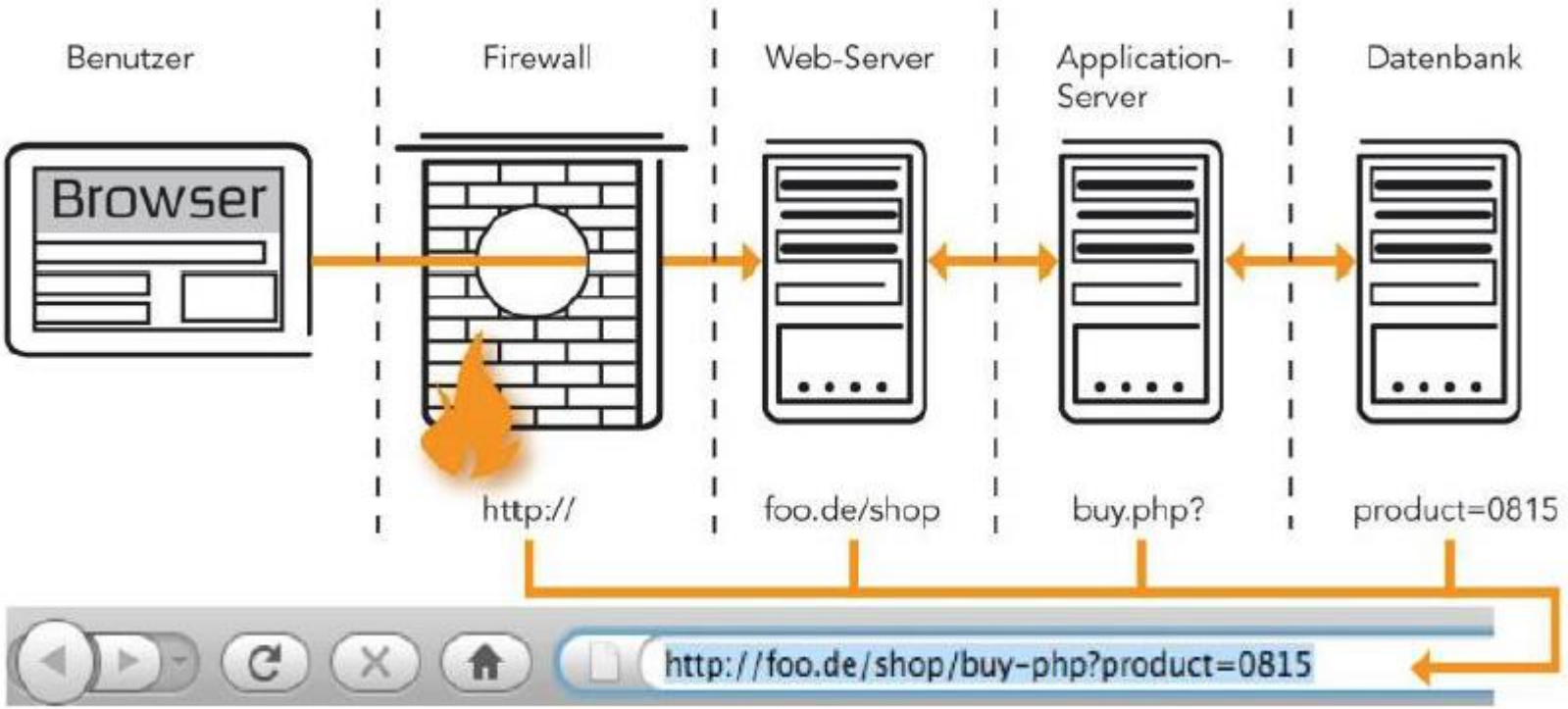
2 Tier Architektur



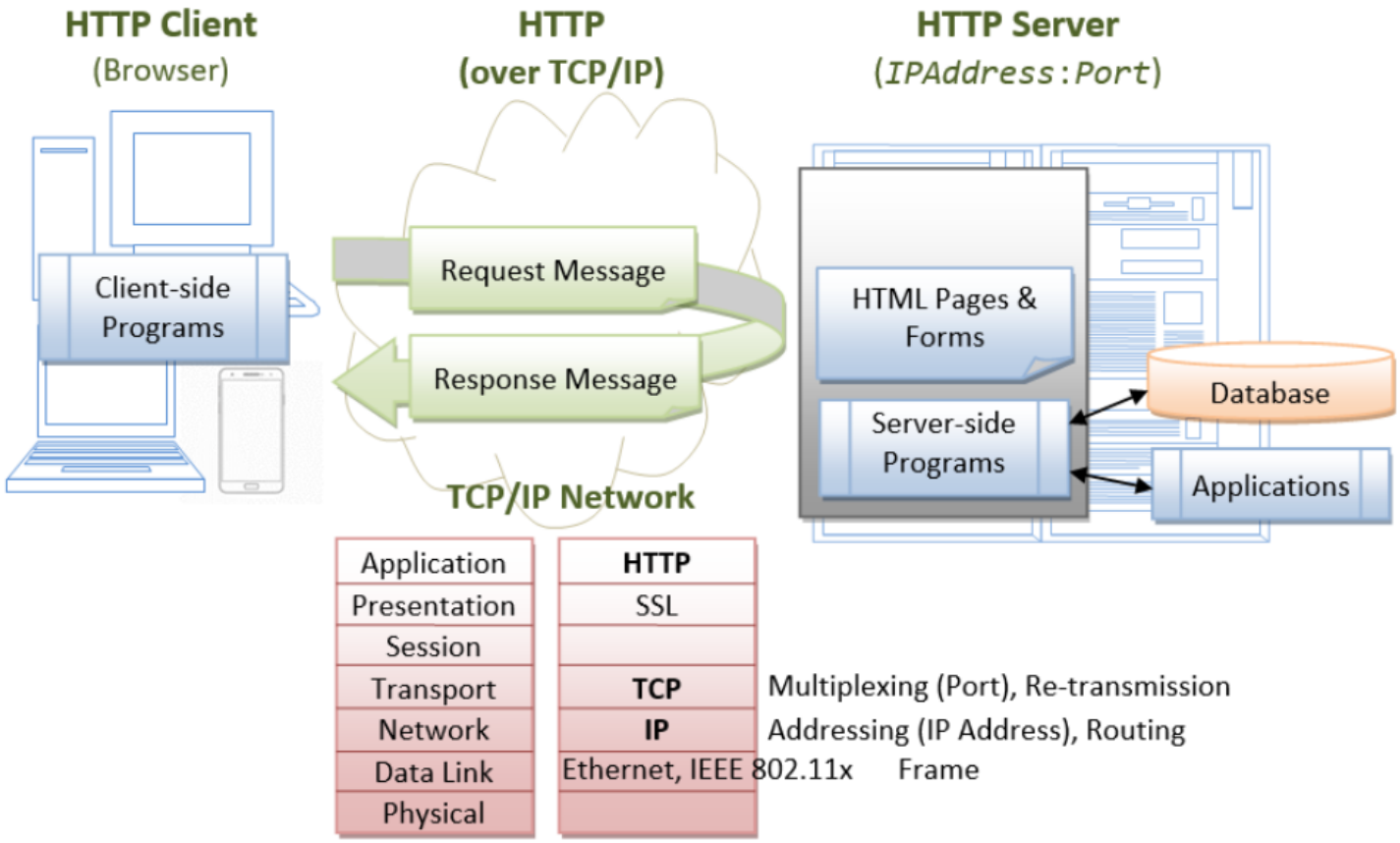
3 Tier Architektur



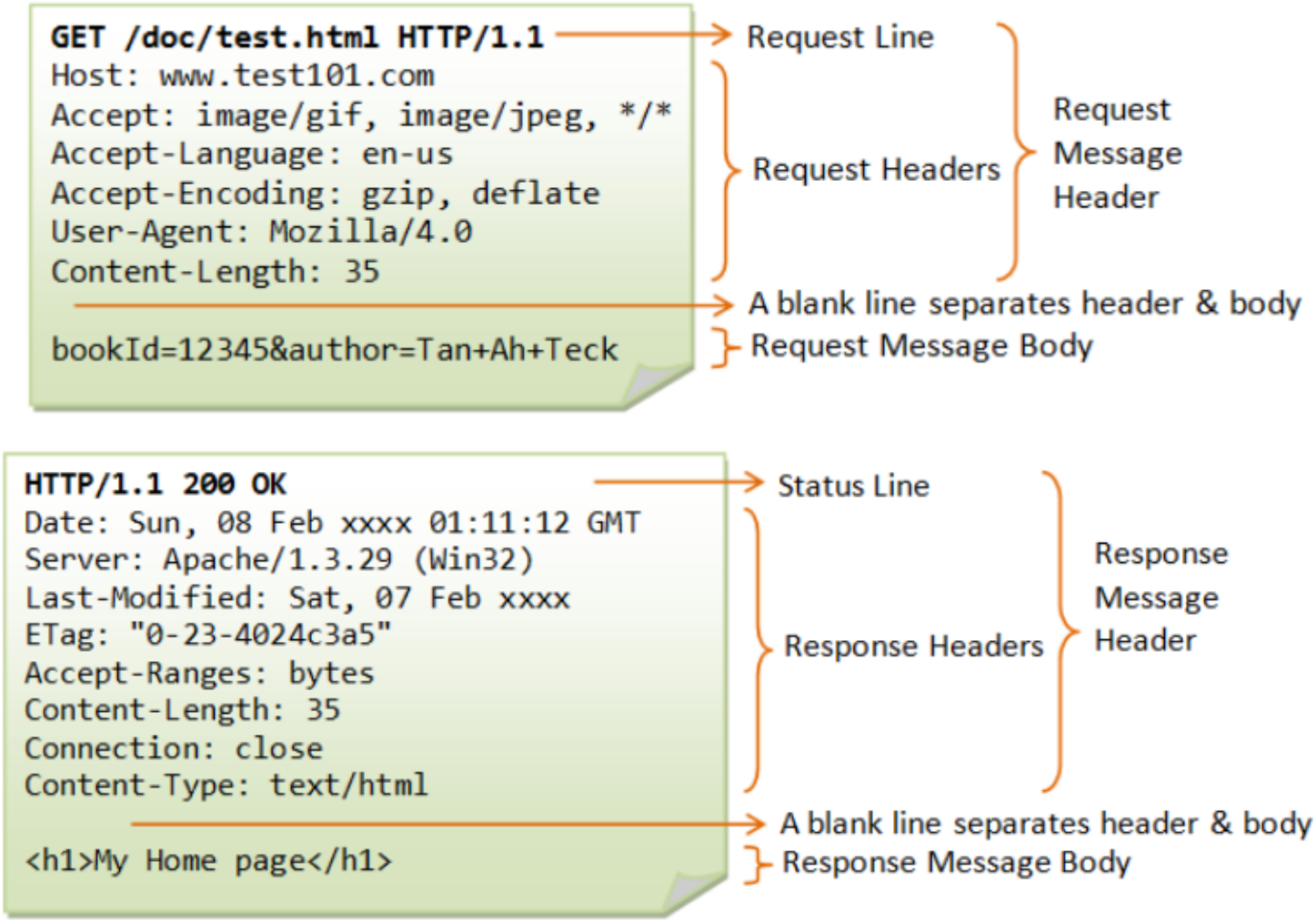
Web-Applikationen



Web-Applikation Ablauf



HTTP-Anfrage



Web-Applikation Ablauf

Die Kommunikationseinheiten in HTTP zwischen Client und Server werden als *Nachrichten* bezeichnet

- Request
- Respons

Nachricht besteht dabei aus zwei Teilen

- Nachrichtenkopf (Header)
- Nachrichtenrumpf (Body)

HTTP-Anfragemethoden

HTTP-Anfragemethoden [Bearbeiten | Quelltext bearbeiten]

GET

ist die gebräuchlichste Methode. Mit ihr wird eine Ressource (zum Beispiel eine Datei) unter Angabe eines URI vom Server angefordert. Als Argumente in dem URI können auch Inhalte zum Server übertragen werden, allerdings soll laut Standard eine GET-Anfrage nur Daten abrufen und sonst keine Auswirkungen haben (wie Datenänderungen auf dem Server oder ausloggen). Die Länge des URIs ist je nach eingesetztem Server begrenzt und sollte aus Gründen der Abwärtskompatibilität nicht länger als 255 Bytes sein. *(siehe unten)*

POST

schickt unbegrenzte, je nach physischer Ausstattung des eingesetzten Servers, Mengen an Daten zur weiteren Verarbeitung zum Server, diese werden als Inhalt der Nachricht übertragen und können beispielsweise aus Name-Wert-Paaren bestehen, die aus einem HTML-Formular stammen. Es können so neue Ressourcen auf dem Server entstehen oder bestehende modifiziert werden. POST-Daten werden im Allgemeinen nicht von Caches zwischengespeichert. Zusätzlich können bei dieser Art der Übermittlung auch Daten wie in der GET-Methode an den URI gehängt werden. *(siehe unten)*

HEAD

weist den Server an, die gleichen HTTP-Header wie bei GET, nicht jedoch den Nachrichtenrumpf mit dem eigentlichen Dokumentinhalt zu senden. So kann zum Beispiel schnell die Gültigkeit einer Datei im Browser-Cache geprüft werden.

PUT

dient dazu, eine Ressource (zum Beispiel eine Datei) unter Angabe des Ziel-URIs auf einen Webserver hochzuladen. Besteht unter der angegebenen Ziel-URI bereits eine Ressource, wird diese ersetzt, ansonsten neu erstellt.

PATCH

Ändert ein bestehendes Dokument ohne dieses wie bei PUT vollständig zu ersetzen. Wurde erst später durch RFC 5789 als Erweiterung des Standard vorgeschlagen (noch nicht verabschiedet).

DELETE

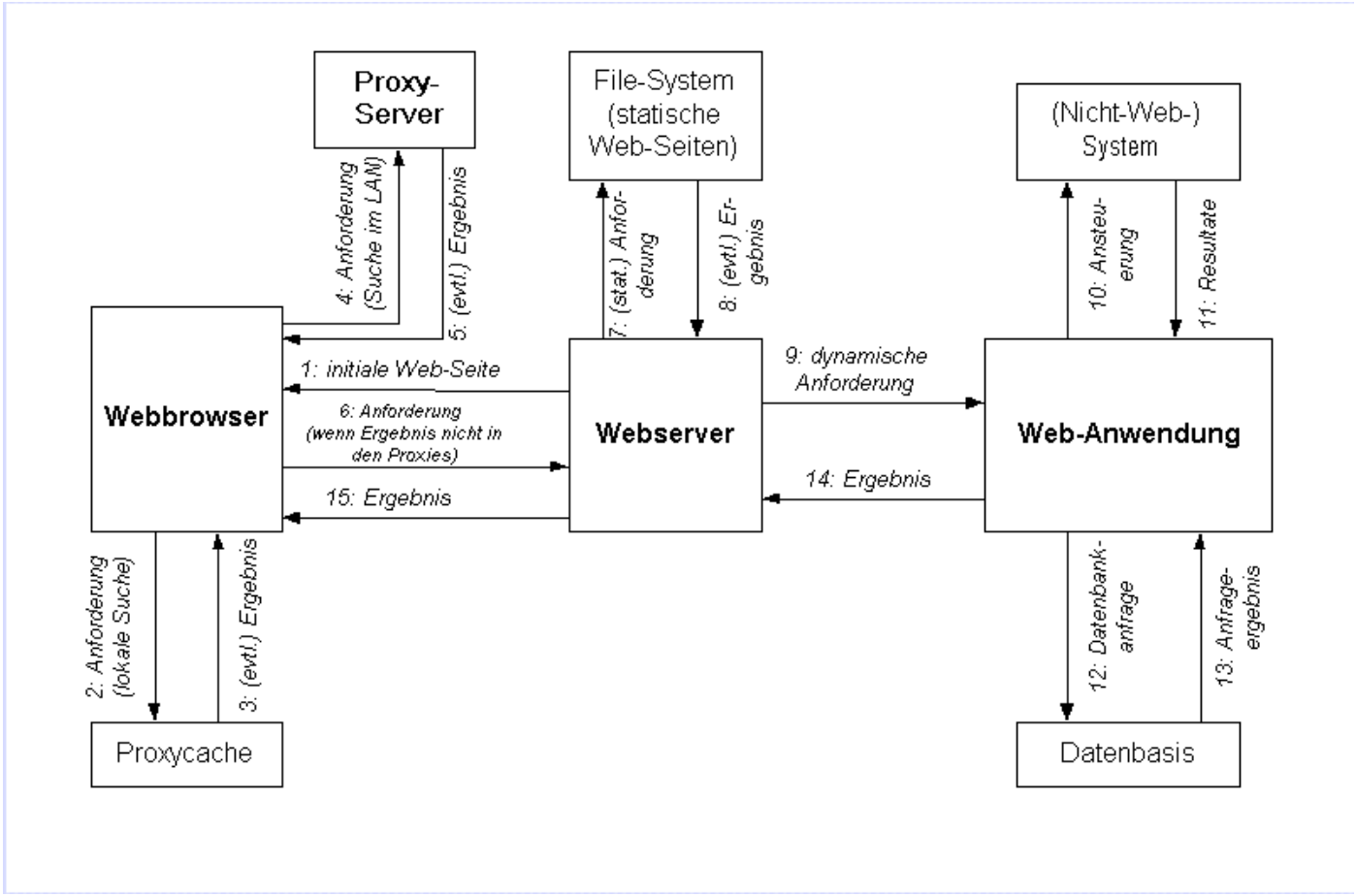
löscht die angegebene Ressource auf dem Server.

Webservices

RESTful Web Services verwenden die unterschiedlichen Anfragemethoden zur Realisierung von Webservices. Insbesondere werden dafür die HTTP-Anfragemethoden GET, POST, PUT/PATCH und DELETE verwendet.

WebDAV fügt die Methoden *PROPFIND*, *PROPPATCH*, *MKCOL*, *COPY*, *MOVE*, *LOCK* und *UNLOCK* zu HTTP hinzu.

Web-Applikation Ablauf



Verbindung mit Datenbank

```php
$id = "root";
$pw = "";
$host = "localhost";
$database = "test";
$table = "artikell";
$meldung = "";
$script=$_SERVER['SCRIPT_NAME'];
print_r($script);
```

```php
$link = mysqli_connect ($host, $id, $pw) or die ("cannot connect");
mysqli_select_db($link, $database) or die ("cannot select DB");
```

Datenbankzugriffe mit php

```php
$id = "root";
$pw = "";
$host = "localhost";
$database = "test";
$table = "artikell";
$meldung = "";
$script=$_SERVER['SCRIPT_NAME'];
print_r($script);
```

```php
$link = mysqli_connect ($host, $id, $pw) or die ("cannot connect");
mysqli_select_db($link, $database) or die ("cannot select DB");
```

Datenbankzugriff mit php

```
mysqli_query($link,"update $table set artnr = $artnr, titel = '$titel', preis = '$preis', inhalt =
'$inhalt' where nr = '$nr'");
$meldung = "Der Artikel wurde upgedated.";
```

Besser mit separatem sql-Statement

```
$sql = "insert into $table (titel, artnr, preis, inhalt) VALUES('$titel', '$artnr', '$preis', '$inhalt')" ;
print_r($sql);
mysqli_query ($link, $sql);
```

php in HTML

```
<table>
    <form action= <?php echo $_SERVER['PHP_SELF']; ?> method= "post">
    <input type=hidden name=action value="save">
    <input type=hidden name=nr VALUE="<? echo $nr ?>">
<tr>
<td>Art.-Nr.</td>
<td><input type=text name="artnr" value="<?php echo $artnr ?>"></td>
</tr><tr>
<td>Titel</td>
<td><input type=text name="titel" value="<?php echo $titel ?>"></td>
</tr><tr>
<td>Preis</td>
<td><input type=text name="preis" value="<?php echo $preis ?>"></td>
</tr><tr>
<td>Text</td>
<td><textarea name="inhalt"><?php echo $inhalt ?></textarea><td>
</tr><tr>
</tr> </td>
<td><input type=submit value="Artikel Updaten"></form></td>
</tr>
</table><p>
```

HTML in php

```
echo '<table>';
echo '<form action="'.$_SERVER['PHP_SELF'].'" method="post">
  <input type="hidden" name="action" value="save">
  <input type="hidden" name="nr" value=' . $nr . '>
<tr>
<td>Art.-Nr.</td>
<td><input type="text" name="artnr" value="' . $artnr.'"></td>
</tr><tr>
<td>Titel</td>
<td><input type=text name="titel" value=' . $titel .'></td>
</tr><tr>
<td>Preis</td>
<td><input type=text name="preis" value=' . $preis .'></td>
</tr><tr>
<td>Text</td>
<td><textarea name="inhalt">' .  $inhalt . '</textarea><td>
</tr><tr>
</tr> </td>
<td><input type=submit value="Artikel Updaten"></form></td>
</tr>
</table><p>';
```