Web

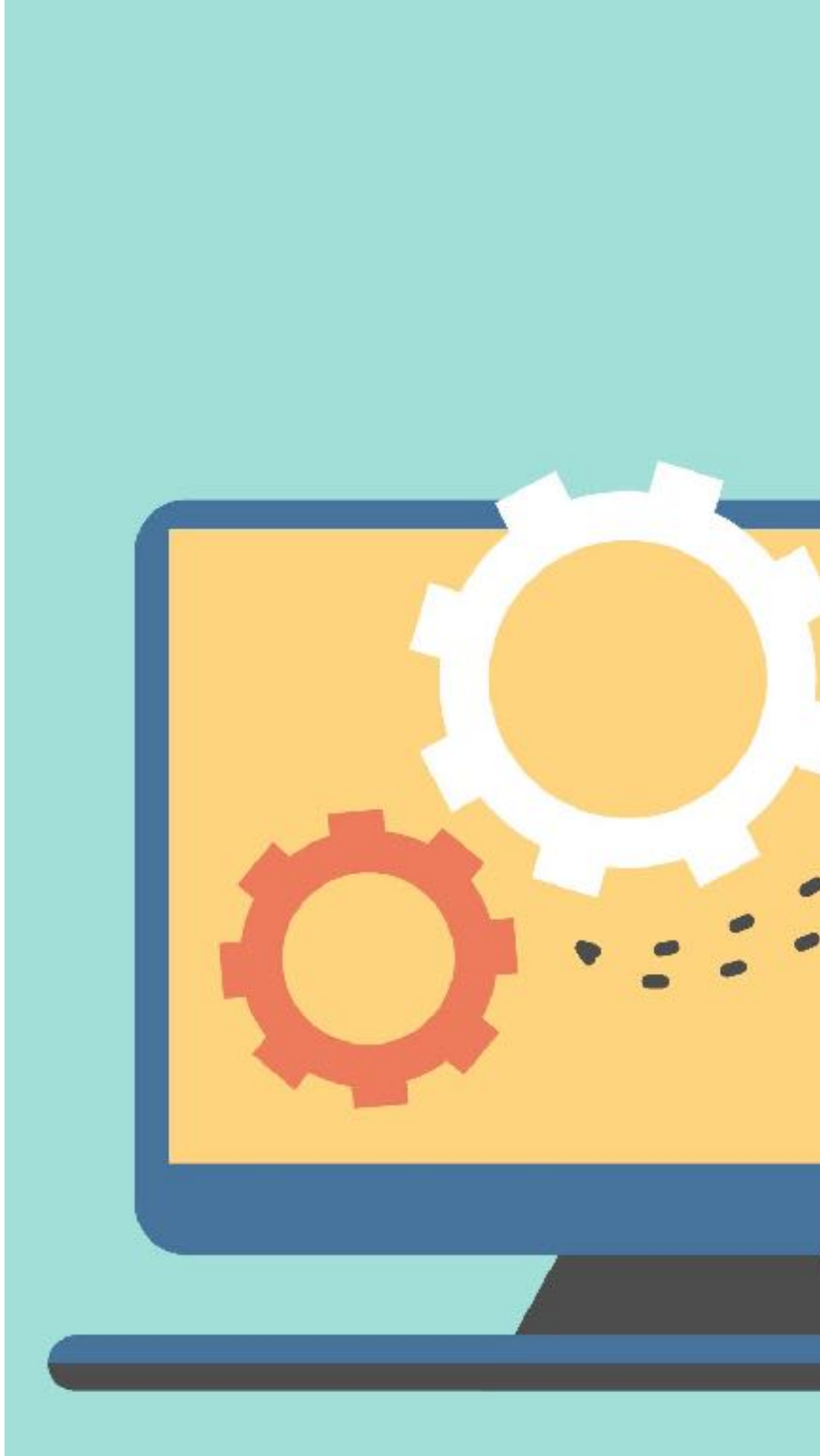
Design & Development

GRUNDLAGEN DER WEBENTWICKLUNG

Tutorium 8.11.2016

THEORIETEIL

Übung 2 & Übung 3



THEORIETEIL: ÜBUNG2

- Apache implementiert das HTTP Protokoll als Grundlage für Kommunikation. Stellt Dokumente in Rechnernetzen bereit —> weit verbreitetster Webserver
- Name aus Respekt vor den amerikanischen Ureinwohnern, oftmals die Legende „a patchy server“
- Es gibt mehrere aktive Versionen —> Unterscheiden sich z.B. in Struktur und unterstützten Modulen, für Unternehmen/ Projekte weiterhin Support wichtig
- Updates sind besonders wichtig, da Apache sehr bekannt —> Automatisierte Prüfung auf Schwachstellen möglich, daher große Gefahr für ein Rechnernetz

THEORIETEIL: ÜBUNG2

- Konfiguration des Apache über httpd.conf im conf Verzeichnis (Aufteilung in kleiner Konfigurationsdateien möglich, einbinden per include)
- Direktive: Verhaltensanweisung, die in der Konfigurationsdatei angegeben werden kann. Beispielsweise <Directory> Direktive
- DocumentRoot: Ist das Wurzelverzeichnis des Apache, dort werden die abzurufenden Dokumente abgelegt. Standardwert: htdocs im Installationsverzeichnis
- Achtung: Alle Dateien unterhalb des DocumentRoot sind über das Internet erreichbar (sofern nicht durch Direktive verboten)

THEORIETEIL: ÜBUNG2

- Verzeichnisschutz: IP-Schutz oder Benutzername/Passwort Schutz
- Beide Varianten in der <Directory> Direktive mithilfe der Require Direktive (Bsp: Require ip 134.2.2.38)
- Nutzername Passwortschutz über Direktive AuthBasicProvider und AuthUserFile in Kombination mit Require (Bsp: Require user valid-user)
- Modul: „Abgekapselte“ Funktionalität, je mehr Module desto langsamer und Fehleranfälliger (Security) wird der Apache Server

THEORIETEIL: ÜBUNG2

- Statische Module: werden mit den Source des Apache kompiliert —> schnell. Bei Änderung muss gesamter Apache neu kompiliert werden
- Dynamische Module: Unabhängig von Apache kompiliert, bei Änderung muss nur das entsprechende Modul neu kompiliert werden. Nachteil: Apache startet pro dynamischen Modul ein neuen Prozess —> langsamer
- Statische Module werden im Makefile angegeben, dynamische in der .conf über LoadModule
- Statische Module: httpd -l

THEORIETEIL: ÜBUNG2

- mod_cgi : Ausführen von CGI-Skripten
- mod_auth : Schützt Inhalte
- mod_status : Statusübersicht
- mod_alias : Ändern von URL-Pfaden
- mod_include : Ermöglicht SSI (Server Side includes)
- Module von Drittanbietern: Nicht von der ASF, z.B. mod_php, mod_perl
- Bedingte Direktive: Direktive mit Bedingung (= D)

THEORIETEIL: ÜBUNG2

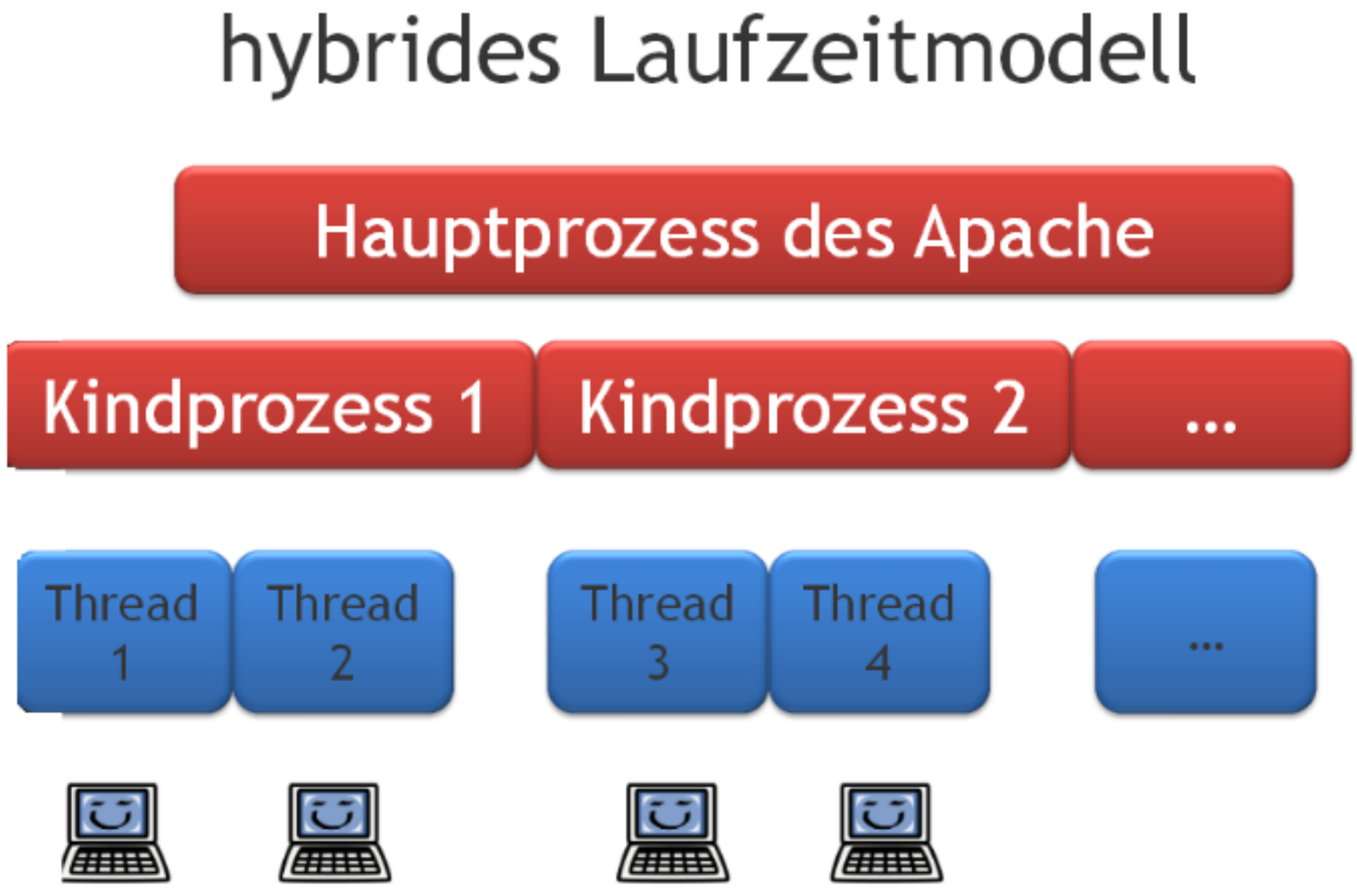


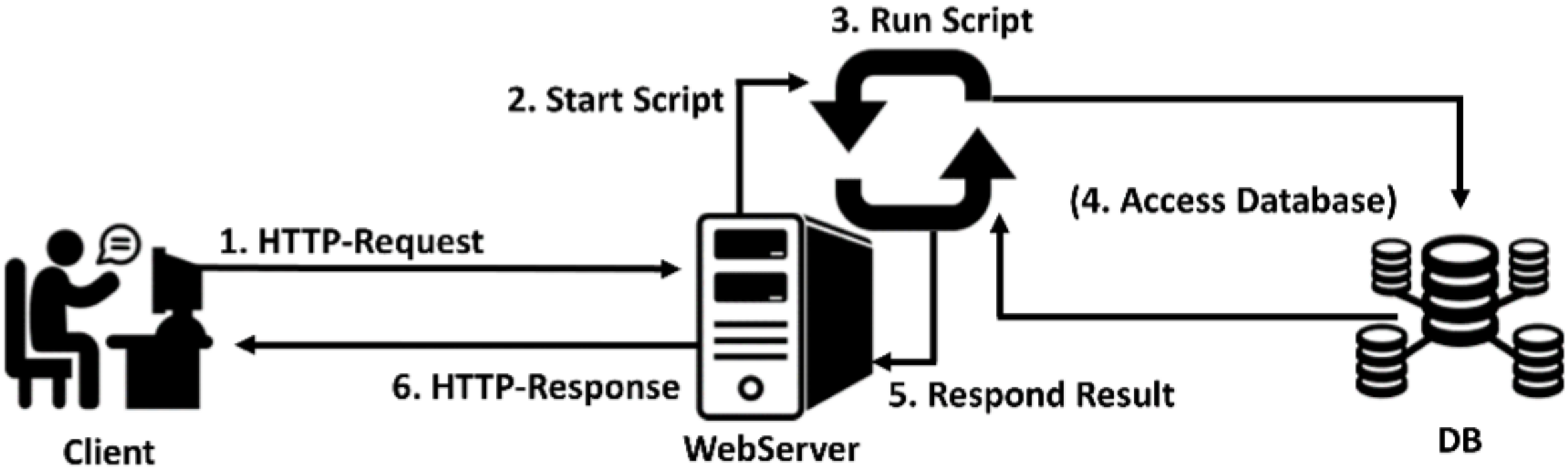
Abbildung 2: Hybrides Laufzeitmodell

THEORIETEIL: ÜBUNG2

- Vorteile Hybrides Laufzeitmodell: Geschwindigkeit (Parallelisierung), Sicherheit (nur Hauptprozess als Root), Stabilität: Absturz eines Threads ist egal

THEORIETEIL: ÜBUNG3

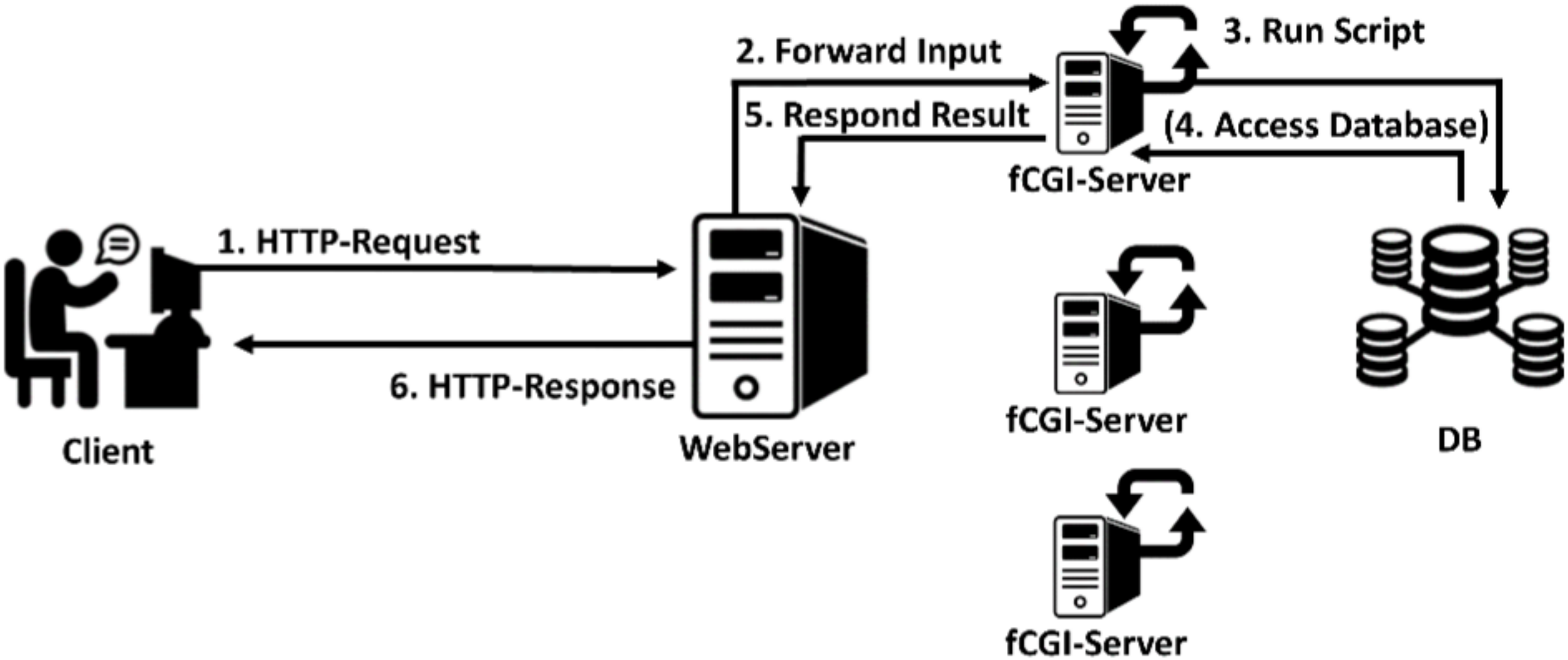
- CGI: Common Gateway Interface, ermöglicht Kommunikation des Apache mit einem Programm / Script auf dem Server



THEORIETEIL: ÜBUNG3

- Nachteil CGI: Trennung von Webserver und Applikationsserver nicht möglich, da Kommunikation über Umgebungsvariablen und Pipes
- Bei jedem Aufruf eines CGI-Scriptes wird ein eigener Prozess erzeugt und jedesmal der jeweilige Interpreter neu geladen —> Langsam
- FastCGI: Ähnlich wie CGI, bei erstem Aufruf von Script wird ein Daemon erzeugt der neue Anfragen auf dieses Script beantwortet
- Dank „Daemonize-it“ Ansatz muss Interpreter nur einmal geladen werden (und Script nur 1x interpretiert werden)

THEORIETEIL: ÜBUNG3 – FASTCGI ABLAUF



THEORIETEIL: ÜBUNG3

- Kommunikation zwischen Webserver und Applikationsserver über TCP Sockets —> physikalische Trennung möglich
- FastCGI ca. 5x schneller als normales CGI
- FastCGI benötigt das Modul mod_fcgid (dynamisch) sowie eine Direktive in der Konfigurationsdatei (Ordner in dem die Script liegen) und Optionen SetHandler fcgid-script und +ExcecCGI (um das Ausführen der Scripte zu erlauben)
- Überführung CGI zu FastCGI: Bibliotheken für fastCGI einbinden und Endlosschleife bauen, die auf neue Anfragen wartet

THEORIETEIL: ÜBUNG3

- PHP vs CGI: PHP Interpreter ist Teil des Webservers (durch mod_php). Somit muss kein neuer Interpreter geladen werden
- Einbinden von PHP: mod_php (dynamisch) + Direktive:

  AddType application/x-httpd-php .php

  (um dem Webserver zusagen, dass bei .php Dateien der Interpreter aufgerufen werden soll)

THEORIETEIL: ÜBUNG3

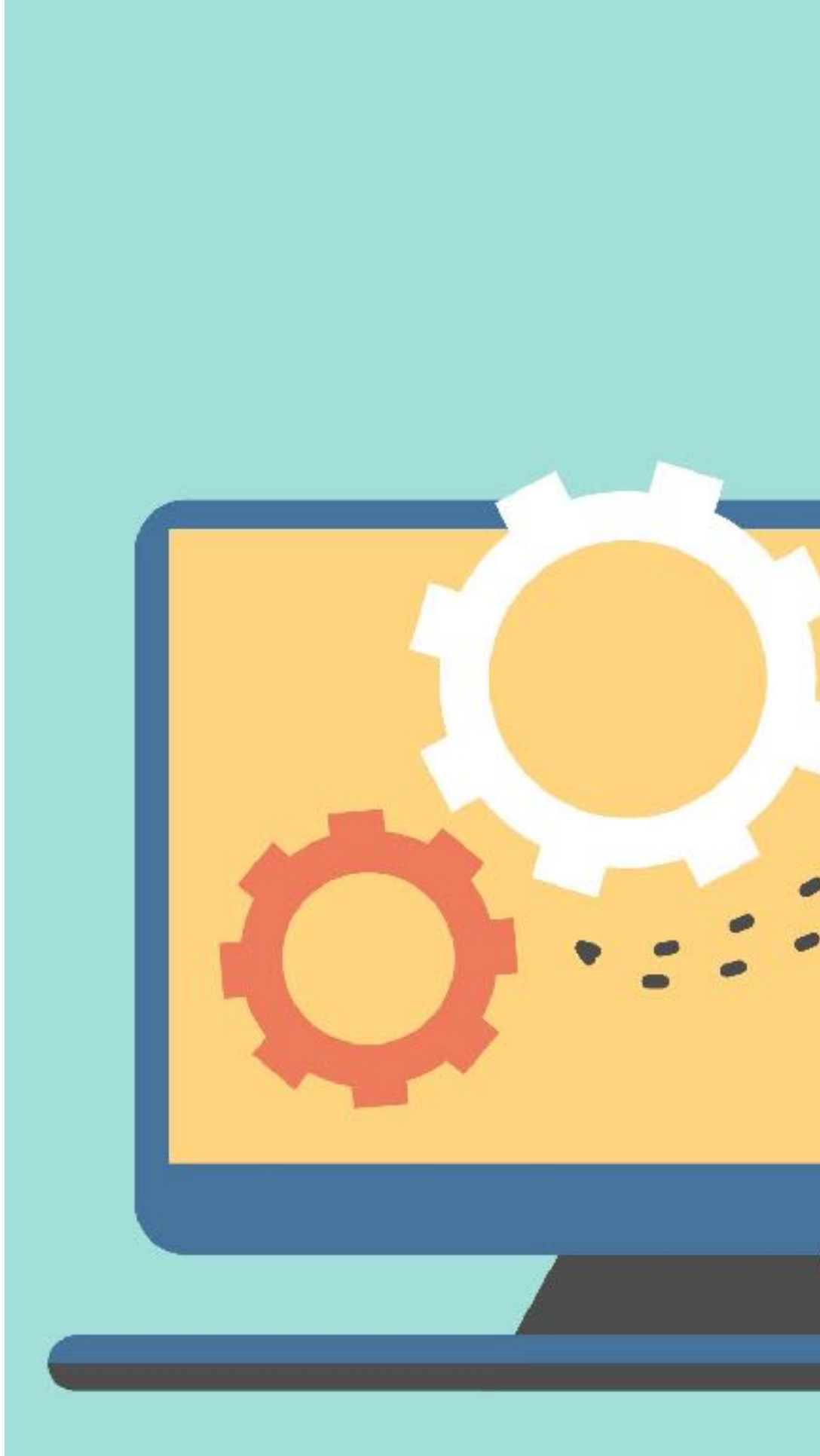
- php.ini: Konfigurationsdatei
- display_errors: Gibt an ob Fehler angezeigt werden sollen
- ingore_user_abort: Gibt an, ob Scriptausführung abgebrochen wird wenn der Client die Verbindung beendet
- include_path: Gibt an in welchen Verzeichnissen nach einzubindenden Dateien gesucht werden sollen
- upload_max_filesize: Maximale Dateigröße die hochgeladen werden darf

THEORIETEIL: ÜBUNG3

- Abbrechen eines Bestellvorgangs:
- Standardmäßig wird die Ausführung des Scripts auf dem Server durch das Abbrechen des Clients gestoppt, somit wird der Bestellvorgang abgebrochen

PRAXIS

Übung 3



MYSQL & PHP

- Tabelle erstellen:

```
mysql> CREATE TABLE test (
    -> ID int NOT NULL AUTO_INCREMENT,
    -> spalte1 VARCHAR(23),
    -> status boolean,
    -> PRIMARY KEY (ID)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

- Werte in die Tabelle eintragen:

```
zxmgf54@infodienste:~$ php -a
Interactive mode enabled

php > $db = new mysqli("localhost", "zxmgf54", "QGPGrqENCebH", "zxmgf54_db");
php > $db->query("INSERT INTO test (spalte1, status) VALUES ('testValue', true)"
);
php >
```

MYSQL & PHP

- Werte aus Tabelle auslesen:

```
php > $result = $db->query("SELECT * FROM test");
php > while($obj = $result->fetch_object()) {
php { echo $obj->spalte1;};
testValue
```