



# 1. Studienprojekt

## Vergleich von CGI und FastCGI

**Name, Vorname:** Knothe, Marvin und Zhang, Yüxi  
**Matrikelnummer:** 640199 und 684163  
**Studienjahrgang:** 2020  
**Fachbereich:** Duales Studium Wirtschaft • Technik  
**Studiengang:** Informatik  
**Modul:** Studienprojekt - 4. Semester  
**Betreuer Hochschule:** Prof. Dr. Arthur Zimmermann  
**Anzahl der Wörter:** 5422

16.08.2022,

*Knothe*

.....  
(Datum/Unterschrift Marvin Knothe)

16.08.2022,

*[Signature]*

.....  
(Datum/Unterschrift Yüxi Zhang)



## **Kurzzusammenfassung**

Das Ziel des vorliegenden Studienprojekts ist der Vergleich der Schnittstellen CGI und FastCGI. Für die Verständlichkeit werden elementare Begriffe von Webanwendungen in einer Nomenklatur erläutert. Im Anschluss wird der Unterschied zwischen statischen und dynamischen Webseiten erklärt. Darauf aufbauend werden das Common Gateway Interface, FastCGI, XAMPP und alternative Technologien beschrieben. In der Anforderungsanalyse werden die Voraussetzungen und Ansprüche an die zu erstellenden Programme erläutert. Um einen Vergleich durchzuführen, wurde eine rudimentäre Webanwendung konzeptioniert und jeweils mit CGI und FastCGI umgesetzt. Dazu wurde eine Grundstruktur in Form einer HTML erstellt, woraus einerseits ein CGI- und andererseits ein FastCGI Skript entstanden sind. Des Weiteren wurden Performance Tests durchgeführt, um die Geschwindigkeit und Auslastung von Arbeitsspeicher und CPU miteinander zu vergleichen. Die daraus folgenden Ergebnisse wurden evaluiert und anhand einer Recherche die Sicherheit der beiden Interfaces miteinander abgeglichen. Zum Schluss wird in einer Fehleranalyse eventuelle Probleme und deren Behandlung betrachtet.

## **Abstract**

The aim of this study project is to compare the CGI and FastCGI interfaces. For the comprehensibility, elementary terms of web applications are explained in a nomenclature. Subsequently, the difference between static and dynamic web pages is explained. Based on this, the Common Gateway Interface, FastCGI, XAMPP and alternative technologies are described. In the requirement analysis, the prerequisites and demands on the programs which shall be created are explained. A rudimentary web application was conceptualized and implemented with CGI and FastCGI to perform the comparison. In addition, a basic structure in form of a HTML was provided, from which in each case CGI and FastCGI script developed. Furthermore, performance tests were performed to compare the speed and utilization of memory and CPU. The following results were evaluated and the security of the two interfaces was compared by means of research. Finally, an error analysis considers possible problems and how to deal with them.



## Inhaltsverzeichnis

<b>I.</b>	<b>Abbildungsverzeichnis .....</b>	<b>II</b>
<b>II.</b>	<b>Tabellenverzeichnis .....</b>	<b>II</b>
<b>III.</b>	<b>Abkürzungsverzeichnis .....</b>	<b>II</b>
<b>1.</b>	<b>Einleitung .....</b>	<b>1</b>
<b>2.</b>	<b>Theoretische Grundlagen von CGI und FastCGI .....</b>	<b>2</b>
2.1	Nomenklatur von Webanwendungen .....	2
2.1.1	HTML.....	2
2.1.2	Perl.....	2
2.1.3	SQL und MySQL.....	2
2.1.4	Webserver.....	2
2.1.5	Skript .....	3
2.1.6	Meta-Variable.....	3
2.2	Vergleich von statischen und dynamischen Webseiten.....	3
2.3	Common Gateway Interface .....	4
2.4	FastCGI.....	6
2.5	XAMPP .....	7
2.6	Alternative Technologien .....	8
2.6.1	ASP.NET .....	8
2.6.2	PHP.....	9
2.6.3	JavaScript .....	9
<b>3.</b>	<b>Anforderungsanalyse .....</b>	<b>10</b>
<b>4.</b>	<b>Konzept.....</b>	<b>11</b>
<b>5.</b>	<b>Implementierung .....</b>	<b>13</b>
<b>6.</b>	<b>Vergleich von CGI und FastCGI .....</b>	<b>16</b>
6.1	Performance.....	16
6.2	Sicherheit.....	17
6.2.1	Sicherheit von CGI-Skripten .....	17
6.2.2	Sicherheit von FastCGI-Skripten.....	18
<b>7.</b>	<b>Fehleranalyse .....</b>	<b>19</b>
<b>8.</b>	<b>Fazit und Ausblick.....</b>	<b>20</b>
<b>9.</b>	<b>Literaturverzeichnis .....</b>	<b>21</b>
<b>10.</b>	<b>Eidesstaatliche Erklärung.....</b>	<b>23</b>



## I. Abbildungsverzeichnis

Abbildung 1: Ablauf einer Generierung einer dynamischen HTML-Datei mittels eines CGI - Skripts.....	4
Abbildung 2: Controllpanel von XAMPP mit Apache Anwendung .....	7
Abbildung 3: localhost Webseite mit Startseite von XAMPP.....	8
Abbildung 4: erstes Konzept für einen rudimentären Webshop mit zwei Produkten (Rechteck), zwei Knöpfen (Ellipsen) und der Warenkorbanzahl (Kreis) .....	11
Abbildung 5: zweites Konzept eines rudimentären Webshops mit zwei Produkten (Rechteck), zwei Textfeldern (abgerundete Rechtecke) und einem Knopf (Ellipse) .....	12
Abbildung 6: Umsetzung des rudimentären Webshops aus Konzept 2, Produkt 1 (links) ist Banane, Produkt 2 (rechts) ist Apfel, darunter sind die jeweiligen Textfelder und der Knopf für den Warenkorb .....	13
Abbildung 7: phpinfo mit der PHP-Version 7.4.28 und der Server API „Apache 2.0 Handler“ als Standardkonfiguration.....	14
Abbildung 8: phpinfo mit der PHP-Version 7.4.28 und der Server API „CGI/FastCGI“ nach der Implementierung.....	15

## II. Tabellenverzeichnis

Tabelle 1: Vergleich von 5 Versuchen (Y-Achse) der Geschwindigkeit in Sekunden (X-Achse) von CGI (blau) und FastCGI (orange).....	16
---	----

## III. Abkürzungsverzeichnis

ASP - Active Server Pages .....	8
CGI - Common Gateway Interface.....	1
CPU - Central Processing Unit.....	10
CSS - Cascading Style Sheets .....	2
HTML - Hypertext Markup Language .....	2
HTTP - Hypertext Transfer Protocol.....	2
PHP - Hypertext Preprocessor .....	9
SQL - Structured Query Language.....	2
TCP - Transmission Control Protocol .....	6
URI - Uniform Resource Identifier .....	4
URL - Uniform Resource Locator.....	16
VM - virtuelle Maschine.....	10
WWW - World Wide Web .....	2



## **1. Einleitung**

Heutzutage begleitet uns das Internet fast überall im Alltag. Im Jahr 2021 nutzten 4,9 Milliarden Menschen das Internet. [LRa] Rund 66,6 Millionen der Einwohner Deutschlands, darunter benutzten 32,4 Millionen Personen das Internet mehrmals täglich und davon 11,6 Millionen sogar fast die ganze Zeit. [Sta22] Ob auf der Arbeit eine E-Mail lesen, eine WhatsApp-Nachricht bekommen oder Musik über das Handy streamen, das Internet ist in der immer weiter digitalisierten Welt kaum wegzudenken. Bei den beliebtesten Webseiten, wie z.B. Google, Facebook oder Amazon, liest der Server passend zu der Anfrage des/der Benutzer\*in Informationen aus einer Datenbank. Daraufhin wird eine dynamisch eine HTML-Seite erstellt, die benutzerspezifische Anzeigen widerspiegelt. [LRa22] Demgegenüber werden bei statischen Seiten, die HTML-Seiten zuerst erstellt und dann vom Webserver angezeigt. [Böh14] (S.244-245)

Eine Technologie, um dynamische Webseiten zu realisieren ist das Common Gateway Interface (CGI) und das davon abstammende FastCGI. Im Rahmen des Studienprojektes der Hochschule für Wirtschaft und Recht Berlin werden beide Varianten unter den Aspekten der Performance und der Sicherheit miteinander verglichen. Jeweils eine Anwendung wird in CGI und FastCGI konzipiert und implementiert. Anschließend werden diese getestet und die daraus entstandenen Daten evaluiert.



## **2. Theoretische Grundlagen von CGI und FastCGI**

### **2.1 Nomenklatur von Webanwendungen**

#### **2.1.1 HTML**

HTML ist die Abkürzung für Hypertext Markup Language und dient der Standardisierung aller Bestandteile einer Webseite. Hypertext bezeichnet die Möglichkeit, Texte mittels Hyperlinks miteinander zu verbinden und ermöglicht das Springen innerhalb dieser. Verlinkte Textdateien können sich auf jedem beliebigen Computer befinden, der Zugang zum Internet hat. Auf diese Weise ist das Informationssystem World Wide Web (WWW) entstanden. Die Beschreibung der einzelnen Komponenten, wie Überschriften, Absätze, Tabellen und Links, funktioniert über Tags, vergleichbar mit Markierungen, welche festlegen, wo eine jeweilige Komponente steht. Mit Cascading Style Sheets, kurz CSS, können auch Bilder, Grafiken, Töne oder Videos in die HTML-Seite eingebunden und nach eigenem Belieben gestaltet werden. [Böh14] (S.86-87)

#### **2.1.2 Perl**

Perl ist eine freie, plattformunabhängige, interpretierte Programmiersprache, die mehrere Programmierparadigmen unterstützt. Sie kann auch genutzt werden, um Skripte zu erstellen.

[Per]

#### **2.1.3 SQL und MySQL**

SQL (kurz für Structured Query Language) ist eine Abfragesprache für den Umgang mit Datenbanken. MySQL ist ein Datenbankmanagementsystem, das für die Verwaltung von Datenbanken auf einem Webserver zuständig ist. [Böh] (S.578)

#### **2.1.4 Webserver**

Ein Webserver bezeichnet ein Programm, welches über das Hypertext Transfer Protocol (HTTP) Daten, meist Dokumente in Form von HTML-Seiten, an Programme vom Client ausliefert. Zudem kann es Skripte aufrufen, um Anfragen vom Client zu bedienen. Im übertragenen Sinn wird auch die Hardware bzw. der Computer, auf dem das Webserver-Programm läuft, als Webserver bezeichnet. [Mün02] (S. 10)



### 2.1.5 Skript

Ein Skript ist Software, welche vom Server passend zu ihrem Interface aufgerufen wird. Es muss sich nicht um ein eigenständiges Programm handeln, sondern kann eine dynamisch geladene oder gemeinsam genutzte Bibliothek oder ein Unterprogramm in einem Server sein. Häufig wird unter dem Begriff "Skript" ein Satz von Anweisungen verstanden, welcher zur Laufzeit interpretiert wird. Dies ist aber nicht allgemein gültig. [Coa04] (S.4)

### 2.1.6 Meta-Variable

Eine Meta-Variable ist ein Parameter, welcher Informationen vom Server an das Skript überträgt. Sie ist nicht immer eine Variable in der Umgebung des Betriebssystems, obwohl dies häufig der Fall ist. [Coa04] (S.4)

## 2.2 Vergleich von statischen und dynamischen Webseiten

Grundlegend gibt es statische und dynamische Webseiten. Dabei ist zu beachten, dass sich die Bezeichnungen statisch und dynamisch nicht auf den Inhalt der Seite beziehen, sondern auf die Art der Generierung der Webseite.

Statische Webseiten sind vorkonfiguriert und benötigen für jede Anfrage jeweils eine eigene HTML-Datei. Nachteilhaft an dem System ist, dass einer einzelnen Webseite viele Dateien zugrunde liegen. Daher sind statische Webseiten eher für Projekte mit kleinem Umfang, wie zum Beispiel Informationsseiten, geeignet. Sie sind weniger komplex als dynamische Webseiten, da sie meist nur aus HTML, JavaScript und CSS bestehen, woraus kürzere Ladezeiten resultieren. Sie erfordern dadurch weniger Konfigurations- und Administrationsaufwand, worunter auch geringere Sicherheitsmaßnahmen fallen. Im Vergleich zu dynamischen Webseiten benötigen sie auch weniger leistungsfähige Server-Hardware, wodurch die Kosten gesenkt können. [Böh14] (S.244-245)

Die meisten modernen Seiten, die im Webbrowser angezeigt werden, wie zum Beispiel Google oder Facebook, sind dynamisch. Im Gegensatz zu statischen Webseiten werden dynamische Webseiten bei jeder Anfrage neu generiert. Die Webseite wird also aus einzelnen Elementen bzw. deren Instruktionen gebaut, wobei der Interpreter auch Daten aus einer Datenbank auswerten kann. Um diese Webseiten zu erstellen, werden Skriptsprachen, wie PHP, JavaScript, Perl oder Python, genutzt. Es existieren zudem verschiedene Technologien, um dynamische Webseiten zu erstellen. Der Fokus dieser Arbeit liegt bei CGI und FastCGI, wobei einige Alternativen kurz erläutert werden. Mithilfe einer der genannten Skriptsprachen kann ein Skript geschrieben werden, welches die CGI Technologie nutzt, um dynamisch eine Webseite zu generieren. [Büh21] (12.05.2022) (S.76)

### 2.3 Common Gateway Interface

Das Common Gateway Interface (CGI) ist eine Schnittstelle, um externe Programme, Software oder Gateway plattformunabhängig auszuführen. Diese Technologie ist seit 1993 in Benutzung, unterstützt Webserver (HTTP Server) und ist kostenlos für Windows und Linux verfügbar. [Coa04] (S.3-4) Die Programme, die mit dem Webserver kommunizieren, werden CGI-Skripte genannt, da sie meistens mit Skriptsprachen erstellt werden. [Mün02] (S.95) Prinzipiell ist dies in fast jeder Programmiersprache möglich. Aufgrund der Leistungsfähigkeit wird oft Perl verwendet. [Böh] (S.579) Voraussetzungen sind dabei einige Standardkonzepte von Unix (stehen unter Windows und anderen Systemen auch zur Verfügung). Die Umgebungsvariablen, Kommandozeilenparameter, Standardeingabe und -ausgabe bilden zusammen die Kommunikationskanäle zwischen dem CGI-Skript und dem Webserver. Parameter werden über Umgebungsvariablen an den Webserver und Requests wie PUT- oder POST-Requests, über den Standardeingabe-Kanal übertragen. Für viele Programmiersprachen gibt es CGI-Bibliotheken, die spezielle Funktionen für die Verarbeitung von Variablen zur Verfügung stellen. [Mün02] (S.95)

Das CGI erlaubt dem Webserver und dem Skript, auf Client-Anfragen zu antworten und daraufhin dynamisch eine HTML-Datei zu generieren. Ein Anwendungsfall ist zum Beispiel das Verarbeiten eines Formulars (siehe Abbildung 1, S.4).

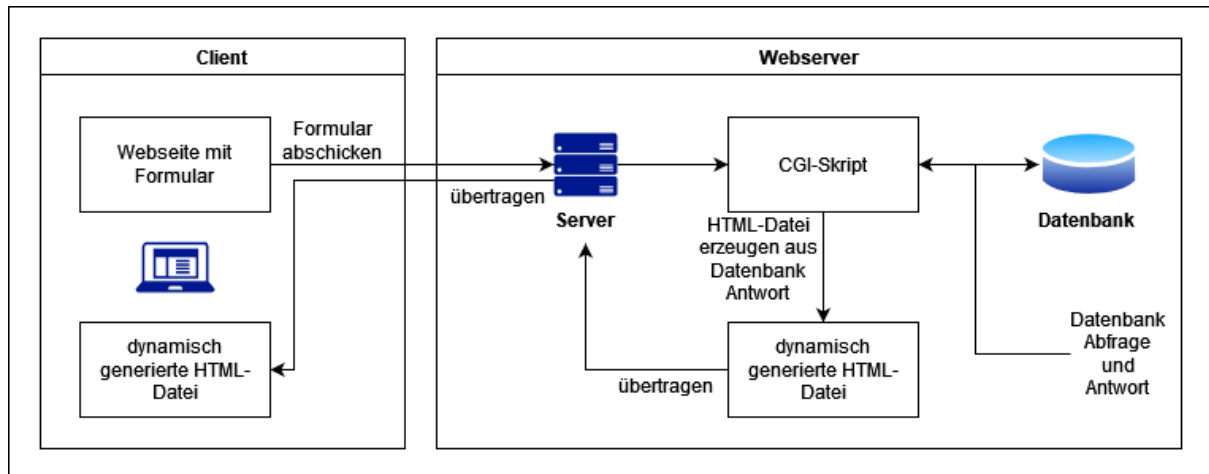


Abbildung 1: Ablauf einer Generierung einer dynamischen HTML-Datei mittels eines CGI-Skripts

Die Client-Anfrage umfasst einen Uniform Resource Identifier (URI), eine Anfragemethode und verschiedene Zusatzinformationen, welche vom Transportprotokoll bereitgestellt werden. Es werden abstrakte Parameter benutzt, die Informationen vom Server zum Skript transportieren und Client-Anfragen beschreiben. Diese Parameter werden auch Meta-Variablen genannt. CGI nutzt Ein- und Ausgabekanäle (z.B. stdout, stdin) um die Informationen anzunehmen und zu verschicken. Zusammen mit einer Programmierschnittstelle wird damit ein



plattformunabhängiger Kontaktpunkt zwischen den Skripten und dem Webserver geschaffen. [Coa04] (S.3-4)

Der Webserver dient dabei als Anwendungs-Gateway und empfängt Anfragen vom Client. Daraufhin wird ein CGI-Skript gesucht, um die Anfrage zu bearbeiten. Die Client-Anfragen werden in CGI-Requests umgewandelt und das Skript ausgeführt. Die dadurch entstehende CGI-Response wird in eine Antwort für den Client umgewandelt. Bei der Verarbeitung der Client-Anfrage ist der Server für die Implementierung von Protokoll- oder Authentifizierung und Sicherheit auf Transportebene zuständig. Zudem muss der Webserver bestimmte Übersetzungen und Protokoll-Konvertierungen mit den Daten der Client-Anfrage durchführen. Diese werden in der Spezifikation „The Common Gateway Interface (CGI) Version 1.1“ [Coa04] (S.8) genauer erläutert. Darüber hinaus hat der Webserver gegenüber dem Client die Verantwortung, die Netzwerkprotokolle einzuhalten, auch wenn das CGI-Skript damit nicht übereinstimmt. Wendet ein Server eine Authentifizierung auf die Anfrage an und diese passiert nicht alle definierten Zugangskontrollen, so kann das Skript nicht ausgeführt werden. [Coa04] (S.7)

Das CGI-Skript bearbeitet Anwendungsanfragen wie zum Beispiel den Datenzugriff und die Dokumentenverarbeitung. Der Webserver bestimmt auf Basis einer vom Client bereitgestellten generischen URI, welches Skript angewendet wird. Diese URI enthält einen hierarchischen Pfad, dessen Komponenten mit einem Schrägstrich („/“) getrennt werden. Für jede Anfrage verknüpft der Server alle oder nur den führenden Teil des Pfades mit einem Skript und platziert dieses an einem bestimmten Punkt in der Pfadhierarchie. Der restliche Teil des Pfades ist, falls vorhanden ein Ressourcen- oder Subressourcenbezeichner, der vom Skript interpretiert wird. Weitere Informationen über die Aufteilung des Pfades stehen in der Spezifikation (3.3 The Script URI [Coa04]) zur Verfügung. [Coa04] (S.8)

Die CGI-Skripte werden seitens des Servers abgearbeitet und lassen sich nicht in HTML-Dateien einbinden. Sie liegen in einem speziellen Verzeichnis auf dem Server namens „cgi-bin“. [Böh] (S.579)

Das Skript wird auf eine vom System vorgegebene Weise, standardmäßig als ausführbares Programm, aufgerufen. Der Server bereitet die CGI-Anfrage vor, welche die Metavariablen (stehen dem Skript direkt nach Ausführung zur Verfügung) der Anfrage und die Daten der Anforderungsnachricht umfasst. Die Anfragedaten müssen dem Skript nicht sofort zur Verfügung stehen, da es auch ausgeführt werden kann, bevor der Server alle Daten vom Client erhält. Das CGI-Skript kann auf eine Datenbank-Abfrage und Antwort reagieren, woraufhin mit den gegebenen Daten vom CGI-Skript eine HTML-Datei dynamisch generiert und zurück zum Client geleitet wird. Bei einem Fehler kann der Server das Skript jederzeit unterbrechen. Dieser Fall kann bei einem Übertragungsfehler zwischen Server und Client eintreten. Das Skript sollte also eine Unterbrechung handhaben können. [Coa04] (S.9)



## 2.4 FastCGI

FastCGI ist eine kostenlose Erweiterung von CGI. Im Gegensatz zu konventionellen Implementierungen von CGI, die einen Anwendungsprozess aufbauen, auf eine Anfrage reagieren und ihn dann beenden, sind FastCGI darauf ausgelegt, langlebige Anwendungsprozesse, d.h. Anwendungsserver zu unterstützen. Eine FastCGI-Applikation läuft nach einer Webserver Anfrage weiter und wartet auf neue Anfragen des Webserver. Dadurch ist FastCGI im Vergleich zu CGI performanter.

Zu Beginn eines FastCGI-Prozesses besteht keine Verbindung, da keine stdin, stdout, stderr Kanäle existieren und nur wenige Informationen über Umgebungsvariablen übergeben werden. [Bro96]

Über einen Listening-Socket oder TCP (Transmission Control Protocol) nimmt der Prozess Verbindungen mit dem Webserver auf. Bei der Nutzung von TCP ist es auch möglich, Webserver und FastCGI auf verschiedenen Rechnern laufen zu lassen, da sie über das Netz kommunizieren. [Mün02] (S.104-105)

Nachdem der FastCGI-Prozess die Verbindung akzeptiert hat, wird ein einfaches Protokoll ausgeführt, um Daten zu empfangen und zu senden. Das Protokoll führt einzelne Multiplex Transportverbindungen zwischen mehreren unabhängigen FastCGI-Anfragen aus. Dies unterstützt Anwendungen, die in der Lage sind, gleichzeitige Anfragen mit Hilfe von ereignisgesteuerten oder Multithreading-Programmiertechniken zu verarbeiten. Des Weiteren stellt das Protokoll innerhalb jeder Anfrage mehrere unabhängige Datenströme in jede Richtung bereit. Auf diese Weise werden beispielsweise sowohl stdout- als auch stderr-Daten über eine einzige Transportverbindung von der Anwendung zum Webserver geleitet. Bei CGI wird dies mit separaten Pipes realisiert.

Eine FastCGI-Anwendung dient verschiedenen Zwecken. Zuerst als ein Responder, da alle in der Anwendung mit einer HTTP-Anfrage verbundenen Informationen aufgenommen werden und sie daraufhin eine HTTP-Antwort generiert. Zudem ist sie ein Authorizer, da sie Entscheidungen über die Berechtigung oder Nichtberechtigung trifft. Zuletzt dient eine FastCGI-Anwendung auch als ein Filter, bei dem die Anwendung alle mit einer HTTP-Anfrage verbundenen Informationen sowie einen zusätzlichen Datenstrom aus einer auf dem Webserver gespeicherten Datei empfängt und eine "gefilterte" Version des Datenstroms als HTTP-Antwort erzeugt. Das Framework ist erweiterbar, so dass später weitere FastCGI definiert werden können. [Bro96]

## 2.5 XAMPP

XAMPP [Apa] ist ein Open-Source-Programmpaket für die Nutzung eines Webserver. Das X ist ein Platzhalter für das jeweilige Betriebssystem (WAMPP für Windows, LAMPP für Linux und MAMPP für Mac). A steht für den Apache Webserver und M für ein Datenbankmanagementsystem, welches für die Datenverwaltung genutzt wird. Das kann MariaDB oder MySQL sein und Phpmyadmin wird als Administrationstool mitgegeben. Die letzten zwei Buchstaben stehen jeweils für eine Skriptsprache. Je nach Vorliebe kann PHP oder Perl genutzt werden. [Böh14] (S.246-247)

Mit XAMPP können CGI und FastCGI Anwendungen implementiert werden. Zusätzlich bietet XAMPP ein Control Panel, über das auf die inkludierten Services zugegriffen werden kann, um sie beispielsweise zu starten oder zu stoppen (siehe Abbildung 2, S.7).

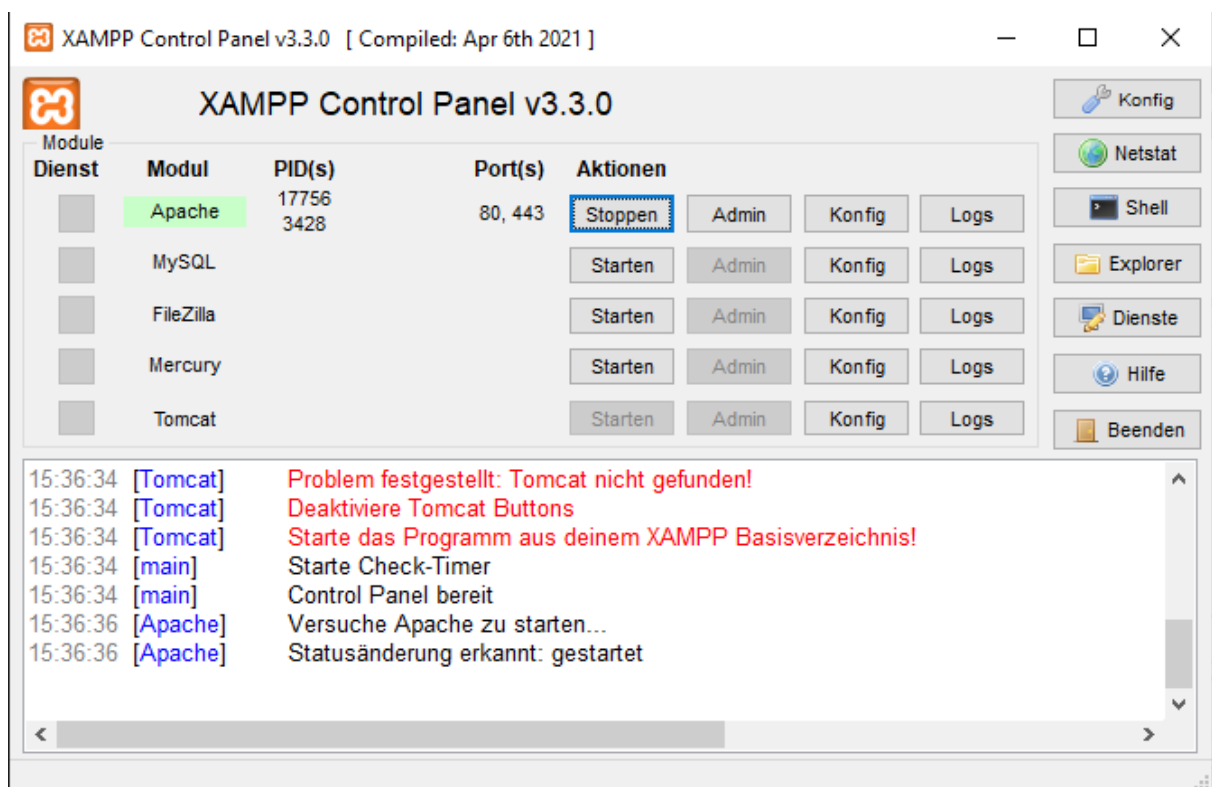


Abbildung 2: Controllpanel von XAMPP mit Apache Anwendung

Sobald der Apache Server gestartet wurde, ist standardmäßig unter "localhost" eine Startseite im Browser verfügbar, bei der sich weitere Informationen finden lassen (siehe Abbildung 3, S.8).

## 2 Theoretische Grundlagen von CGI und FastCGI



Abbildung 3: localhost Webseite mit Startseite von XAMPP

## 2.6 Alternative Technologien

Es gibt viele verschiedene Technologien, um dynamische Webseiten zu erstellen. Einige Alternativen werden im folgenden Kapitel kurz erläutert.

### 2.6.1 ASP.NET

ASP.NET ist die Kurzform von Active Server Pages und gehört der kostenlosen Open Source Software-Plattform „.NET“ an. Es ist eine Webtechnologie bei denen die Webanwendungen bzw. deren Skripte können mit den Programmiersprachen C#, JScript oder Visual Basic Script erstellt werden können. [Böh14] (S.203)

ASP.NET ist kommerziell von Microsoft vertrieben und daher proprietär für entsprechende Microsoft-Webserver (Internet Information Server, Personal Webserver). Der Skriptcode lässt sich direkt in die HTML-Datei einbetten und die Ausführung der Skripte erfolgt serverseitig. [Böh] (S.579)



### 2.6.2 PHP

PHP ist eine Open-Source-Skriptsprache und steht mit einem rekursiven Akronym für Hypertext Preprocessor. [php] Es ist eine serverseitige Technologie, welche unter Linux/Unix oder Windows betrieben werden kann. [Böh14] (S.202)

Ein PHP-Skript lässt sich, wie ASP, direkt in HTML-Dateien einbetten. Der Server erkennt anhand der Dateiendung .php (bzw. .php3, .php41), dass es sich um keine reine HTML-Datei handelt, und übergibt das Skript an den zur Ausführung verantwortlichen PHP-Interpreter. Die PHP-Programmierung ist bei Einsteiger\*innen in der Webprogrammierung sehr beliebt. [Böh] (S. 579)

### 2.6.3 JavaScript

JavaScript findet heutzutage immer mehr an Bedeutung. [Roh18] (S.2) Im Unterschied zu den bisher genannten Technologien wird JavaScript nicht serverseitig, sondern durch den Webbrowser der Anwender\*innen ausgeführt. Es ist also eine clientseitige Skriptsprache, wobei jedoch auch Möglichkeiten zur serverseitigen Ausführung bestehen. JavaScript benötigt deshalb keinen Webserver, sondern kann direkt im Browser getestet werden. [Böh14] (S.203) Hierdurch lassen sich Funktionen realisieren, die mit HTML nicht möglich sind. Ein paar Beispiele hierfür wären eine Fehlermeldung bei einem fehlenden Eintrag in ein Formular oder ein Dialogfenster anzuzeigen. [Böh] (S.579)



### **3. Anforderungsanalyse**

Für den Vergleich von CGI und FastCGI müssen spezifische Kriterien festgelegt und betrachtet werden. Messbare Eigenschaften sind die Geschwindigkeit bei einer Anfrage in Sekunden, die Auslastung der Arbeitsspeichernutzung in Megabyte und die Auslastung des Prozessors (oder auch Central Processing Unit – CPU) in Prozent. Zudem wird die Sicherheit gegenübergestellt, obwohl bei diesem Kriterium sich auf Recherche gestützt wird, da umfangreiche Penetration-Tests zum Überprüfen der Sicherheit den Umfang dieser Arbeit übersteigen. Um die Vergleichbarkeit, zu gewährleisten werden zwei Webseiten unter den gleichen Umständen dynamisch generiert. Die Webseiten sind vom HTML Code identisch. Es werden Anpassungen vollzogen, sodass die Webseiten jeweils mit CGI und FastCGI funktionieren. Beide Skripte werden aufgrund der Leistungsfähigkeit, wie schon in 2.3 Common Gateway Interface beschrieben, mit der Programmiersprache Perl geschrieben. Die Umsetzung erfolgt mit WAMPP, da dadurch alle wichtigen Programme auf einmal installiert werden können. Zudem steht zurzeit kein Linux System zur Verfügung steht. Die Umsetzung mit einer virtuellen Maschine (VM), wie zum Beispiel mit Oracle Virtual Box steht außer Frage, da die Performance stark fluktuieren kann. [Vir22]

## 4. Konzept

Das erste Konzept für die CGI- und FastCGI-Anwendungen stellt einen typischen Anwendungsfall bei dynamischen Webseiten dar. Es handelt sich um die Umsetzung eines rudimentären Webshops (siehe Abbildung 4, S.11).

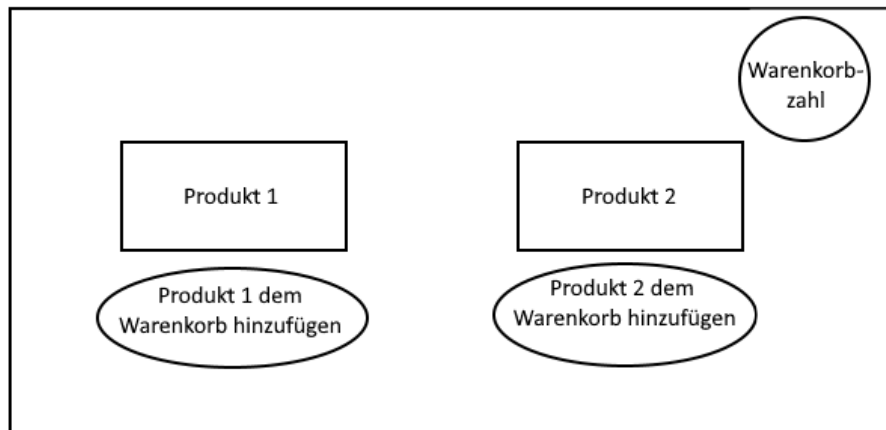


Abbildung 4: erstes Konzept für einen rudimentären Webshop mit zwei Produkten (Rechteck), zwei Knöpfen (Ellipsen) und der Warenkorbanzahl (Kreis)

Es gibt zwei Produkte (Rechteck) mit jeweils einem dazugehörigen Knopf (Ellipse), um das Produkt dem Warenkorb (Kreis) hinzuzufügen. Der Warenkorb ist ein Zähler, welche die angezeigte Zahl nach dem Drücken von einem der beiden Knöpfe um eins erhöht. Da jedoch ein Problem mit der Warenkorbanzahl bei FastCGI bezüglich der Cookies auftrat (siehe 5. Implementierung), wurde ein zweites Konzept erstellt.

#### 4 Konzept

---

Das zweite Konzept ist ähnlich aufgebaut wie das erste Konzept. Jedoch gibt es diesmal keinen Zähler, der erhöht wird, wenn ein Knopf betätigt wird.

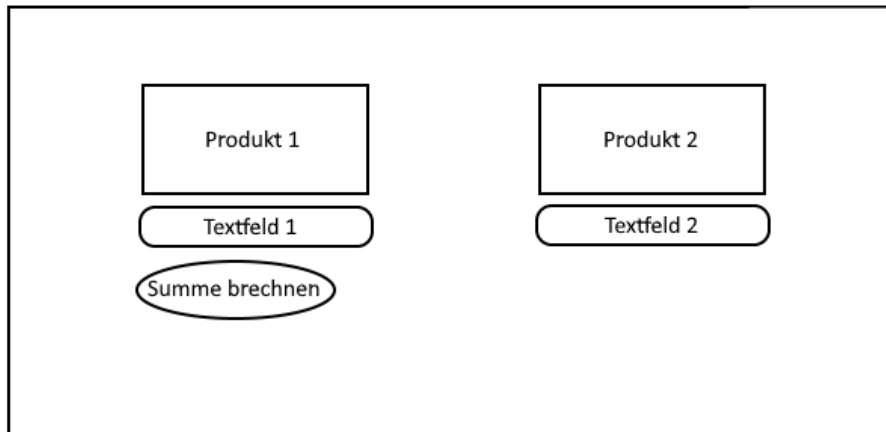


Abbildung 5: zweites Konzept eines rudimentären Webshops mit zwei Produkten (Rechteck), zwei Textfeldern (abgerundete Rechtecke) und einem Knopf (Ellipse)

Anstatt der beiden Knöpfe gibt es zwei Textfelder (abgerundetes Rechteck), in die eine Zahl eingegeben werden kann. Wenn diese Aktion ausgeführt und auf den neuen zusätzlichen Knopf gedrückt wird (Ellipse), um das Skript auszuführen, wird der/die Benutzer\*in weitergeleitet. Dort wird ein kurzer Text eingeblendet, in dem unter anderem die Summe der beiden eingegebenen Zahlen steht. Diese Zahl stellt die Summe der Produkte dar.



## 5. Implementierung

Der erste Schritt beinhaltet zunächst die Installation von XAMPP. Dies dient dazu, die CGI- und FastCGI-Skripte mit der Hilfe eines Apache Servers zu verwenden. Als Entwicklungsumgebung wird die IntelliJ IDEA genutzt, da bereits Erfahrungen mit dieser gesammelt wurden. Bevor die Skripte erstellt werden, wird zunächst eine Grundlage geschaffen, von der die Skripte aufgerufen werden. In diesem Fall ist es ein HTML-Dokument, welches unter anderem zwei Knöpfe besitzt, um dem Warenkorb eine Ware hinzuzufügen. Zusätzlich wird auf Knopfdruck das Skript aufgerufen, welches die Eingaben weiterverarbeitet. Das erste Konzept ist mit CGI umgesetzt, sodass jedes Mal, wenn der Knopf gedrückt wird, die Anzahl der Waren im Warenkorb um 1 erhöht wird. Realisiert wird dies mit der Hilfe von Cookies, sodass die Anzahl gespeichert und wenn benötigt wieder ausgelesen werden kann. Beim Versuch das Konzept mit FastCGI umzusetzen, stellt sich heraus, dass diese Technologie keine Cookies besitzt. Keine gefundene Dokumentation hat diese Thematik behandelt, wodurch keine Lösung für das Problem gefunden werden konnte. Dementsprechend wurde ein anderes Konzept aufgestellt, welches keine Cookies benötigt, um Daten zwischenspeichern. Im zweiten Konzept sind statt der zwei Knöpfe, zwei Textfelder im HTML-Dokument eingebettet. Es besteht die Möglichkeit eine Zahl einzugeben und über einen Knopf das Skript aufzurufen. Die Implementierung mit CGI erfolgt problemlos. Bei FastCGI existiert ein ähnliches Problem wie beim ersten Konzept. Das neue Problem handelt nicht um Cookies, sondern um die Möglichkeit Parameter auszulesen, welche mitgeschickt werden. In den Parametern stehen die Zahlen, welche im HTML-Dokument in den Feldern eingetragen werden. (siehe Abbildung 6, S.13)

### Der Shop



Produkt 1	Produkt 2
	
<input type="text"/>	<input type="text"/>
<input type="button" value="Zum Warenkorb hinzufügen"/>	


Abbildung 6: Umsetzung des rudimentären Webshops aus Konzept 2, Produkt 1 (links) ist Banane, Produkt 2 (rechts) ist Apfel, darunter sind die jeweiligen Textfelder und der Knopf für den Warenkorb

Letztendlich wird sowohl die CGI als auch die FastCGI-Bibliothek verwendet, um das Konzept für FastCGI umzusetzen. Grund dafür ist, dass beide Bibliotheken nicht dieselben Grundfunktionalitäten besitzen, um zum Beispiel Cookies oder Parameter entgegenzunehmen und somit nicht eigenständig die gleiche Funktionalität bieten können. Um die Parameter aus der Anfrage zu empfangen, wird die CGI Bibliothek verwendet und das Skript um FastCGI erweitert. Dieser Weg ist im Nachhinein auch für das erste Konzept möglich, letztlich blieb die Wahl jedoch beim zweiten Konzept.

## 5 Implementierung


Zusätzlich zu den verschiedenen Skripten, muss auch der Apache Server für FastCGI konfiguriert werden. Für FastCGI gibt es ein eigenes Modul, welches auf dem Apache Server installiert werden muss. Es existiert eine Einstellung, welche in der Konfigurationsdatei hinzugefügt wird. Sobald die Schritte korrekt ausgeführt sind, lässt sich dies auch im Browser mittels PHPInfo überprüfen. Dort ist unter anderem ein Eintrag für die verwendete Server API.

PHP Version 7.4.28



System	Windows NT DESKTOP-1NLQI2C 10.0 build 19044 (Windows 10) AMD64
Build Date	Feb 24 2022 02:13:42
Compiler	Visual C++ 2017
Architecture	x64
Configure Command	cscript /nologo /e:js cscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk_shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk_shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	F:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.TS,VC15
PHP Extension Build	API20190902.TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, zip, compress.zlib, compress.bzip2, https, ftps, phar
Registered Stream Socket Transports	tcp, udp, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, zlib.*, bzip2.*

This program makes use of the Zend Scripting Language Engine:  
Zend Engine v3.4.0, Copyright (c) Zend Technologies



### Configuration apache2handler

Apache Version	Apache/2.4.52 (Win64) OpenSSL/1.1.1m PHP/7.4.28
Apache API Version	API20190902

Abbildung 7: phpinfo mit der PHP-Version 7.4.28 und der Server API „Apache 2.0 Handler“ als Standardkonfiguration

## 5 Implementierung

PHP Version 7.4.28	
System	Windows NT DESKTOP-1NLOI2C 10.0 build 19044 (Windows 10) AMD64
Build Date	Feb 24 2022 02:13:42
Compiler	Visual C++ 2017
Architecture	x64
Configure Command	cmd /c "cd /d %~dp0\src\configure && phpize --enable-snapshot-build --enable-debug-pack --with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk\shared --with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk\shared --enable-object-out-dir=.obj --enable-com-dotnet=shared --without-analyzer --with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	F:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902.TS.VC15
PHP Extension Build	API20190902.TS.VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, zip, compress, zlib, compress, bzip2, https, fips, phar
Registered Stream Socket Transports	tcp, udp, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, zlib.*, bzip2.*
This program makes use of the Zend Scripting Language Engine: Zend Engine v3.4.0, Copyright (c) Zend Technologies	

## Configuration

## bcmath

BCMath support	enabled
----------------	---------

Abbildung 8: phpinfo mit der PHP-Version 7.4.28 und der Server API „CGI/FastCGI“ nach der Implementierung

Es ist zu erkennen, dass sich die Server APIs unterscheiden. Wenn das FastCGI Modul nicht installiert ist, wird der Apache 2.0 Handler verwendet (siehe Abbildung 7, S.14) und falls das Modul installiert und konfiguriert ist, wird CGI/FastCGI als Server API verwendet (siehe Abbildung 8, S.15). Die Installation erfolgte nach einem Tutorial von Paul Shipley. [Shi19]

## 6. Vergleich von CGI und FastCGI

### 6.1 Performance

Um Performancetests durchzuführen, wird das Apache Benchmarking Tool verwendet, welches bei der Installation von dem Apache-Webserver mitgegeben wird. Es bietet die Möglichkeit, eine festgelegte Anzahl an Anfragen an eine festgelegte Uniform Resource Locator (URL) zu senden. Der Test wird vom selben Computer ausgeführt, auf dem auch der Webserver läuft. Bei diesem Test handelt es sich um 2000 Anfragen an den Server, wobei 50 Anfragen gleichzeitig versendet werden. Bei den Anfragen handelt es sich um POST-Anfragen, welche zusätzlich 2 Werte beinhalten, die der/den Benutzer\*innen normalerweise in der HTML-Datei selbst angeben werden würde. Der Test wurde sowohl für CGI als auch für FastCGI 5-mal wiederholt.

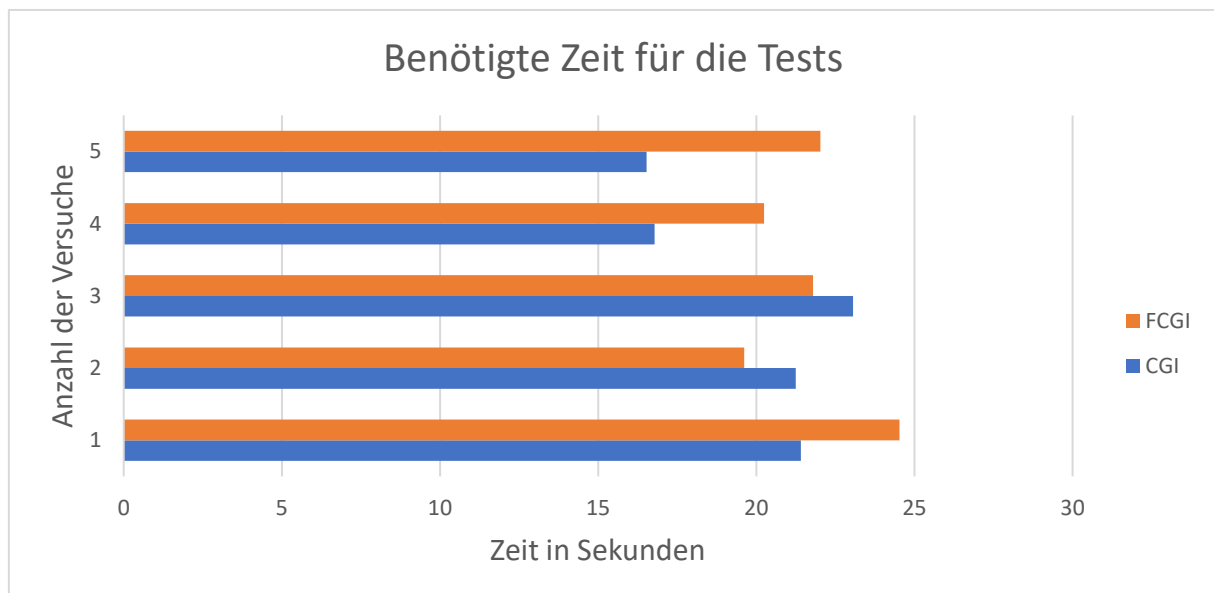


Tabelle 1: Vergleich von 5 Versuchen (Y-Achse) der Geschwindigkeit in Sekunden (X-Achse) von CGI (blau) und FastCGI (orange)

Es ist erkennbar, dass keiner der beiden Technologien die jeweils andere deutlich schlägt. Beide benötigen zwischen 15 und 25 Sekunden. Da zuerst 5-mal CGI und danach FastCGI getestet wurde, lässt sich der erste Test bei FastCGI nicht direkt mit dem Versuch von CGI vergleichen, sondern das Gesamtbild muss betrachtet werden (siehe 7. Fehleranalyse). Dabei lässt sich herauslesen, dass keine der beiden Technologien wesentlich schneller war,

In Bezug auf die Arbeitsspeicherauslastung nutzen beide Interfaces ähnlich viel. Laut dem Windows Task-Manager nutzen beide maximal 100MB der insgesamt verbauten 16GB, wobei die Unterschiede marginal und somit vernachlässigbar sind. Bei der Prozessorauslastung ist



anzumerken, dass sobald der Test gestartet wird, bei dem Test-Computer alle 6 Kerne bzw. 12 logische Prozessoren 100% Auslastung erreichen. Das tritt sowohl bei CGI als auch bei FastCGI ein und somit gibt es auch in dieser Hinsicht keinen Unterschied zwischen den beiden Technologien.

## 6.2 Sicherheit

Das World Wide Web basiert auf dem HTTP (Hypertext Transfer Protocol) und HTML. Informationen werden von einer hohen Anzahl von Webservern bereitgestellt. Der Apache-Webserver ist Open Source und unter der Apache Public Licence kostenlos verfügbar. Im Jahr 2002 wurde Apache mit einem Anteil von deutlich über 50 Prozent am häufigsten genutzt. [Mün02] (S.5) Heutzutage werden auch die meisten aktiven Webseiten mit Apache als Webserver betrieben. [Sta221] Des Weiteren werden Webserver auch für interne Firmennetzwerke (Intranet) genutzt, da sie eine einfache und standardisierte Schnittstelle zwischen Server-Anwendungen und Nutzer\*innen bieten und entsprechende Client-Software (Webbrowser) für jede Betriebssystemumgebung kostenlos verfügbar ist. Da Webserver ein öffentlich zugängliches System darstellen, sollten diese sicher in der Netzwerkumgebung konfiguriert sein. [Mün02] (S.5) Es gibt viele Stellen eines Webserver, die angegriffen werden können und allgemein für Webserver können Maßnahmen ergriffen werden, um diese zu schützen. Eine Authentifizierung des Clients kann über Cookies oder Session-IDs realisiert werden. [Mün02] (S.10) Bei nachfolgenden Anfragen wird das Cookie mitverschickt, wodurch spezifisch nur für die Benutzer\*innen freigegebene oder auch dynamisch erzeugte Ressourcen angezeigt werden. [Mün02] (S.76-77) Zudem sollten alle Programme wichtige sicherheitsrelevante Updates oder Bugfixes einspielen. Nur notwendige Dienste und Programme sind im besten Fall auf dem Webserver installiert. Rechte und Konfigurationsmöglichkeiten sollten für "normale Benutzer\*innen" begrenzt werden, um Schäden zu minimieren. [Mün02] (S.10)

### 6.2.1 Sicherheit von CGI-Skripten

Zusätzliche Software wie CGI-Skripte sind auf sicherheitsrelevante Fehler zu überprüfen, da hierbei oft gravierende Sicherheitsmängel auftauchen. Dies ist darauf zurückzuführen, dass die Programme meist schnell für ein bestimmtes Anwendungsproblem entwickelt wurden. Daher sind sie nicht verbreitet und unzureichend getestet. Eines der Hauptprobleme bei der Nutzung von CGI-Skripten ist die Verarbeitung von Benutzereingaben. Das Skript wertet die Benutzereingaben in ein HTML-Formular, Resultate eines anderen Skripts oder die von dem/der Benutzer\*in angeforderte URL aus. Diese, aber auch andere Daten, die nicht aus einer Benutzereingabe stammen, können manipuliert werden, falls Zugriff darauf besteht. Zum Beispiel betrifft dies die Daten zur Realisierung der Authentifizierung wie Cookies, in die Links



einer HTML-Seite kodierte Daten oder hidden input fields (Formularfelder, welche vom Webserver ausgefüllt werden und nicht angezeigt werden). [Mün02] (S.89)

Die CGI-Skripte werden vom Webserver ausgeführt, daher muss das lokal verwendete Benutzerkonto Lese- und Schreibrechte für das Skript besitzen. In den meisten Fällen ist es nicht erwünscht, dass CGI-Skripte für Webbenutzer\*innen sichtbar werden, da somit ein Zugriff auf den Quellcode des Skripts möglich ist. Dies ist besonders relevant, falls nicht öffentliche CGI-Skripte lokale Passwörter für Datenbankzugriffe im Quelltext speichern. Mithilfe von Wrapper-Programmen wie zum Beispiel suexec, CGIWrap oder sbx kann ein Wechsel des Benutzerkontextes durchgeführt werden, wodurch CGI-Skripte nur vom jeweiligen Eigentümer gelesen und ausgeführt werden. Außerdem wird der Zugriff anderer lokaler Benutzer\*innen verhindert. [Mün02] (S.102-103)

#### 6.2.2 Sicherheit von FastCGI-Skripten

FastCGI-Applikationen, die vom Webserver gestartet werden, können auch Wrapper wie zum Beispiel suexec verwenden, um unter anderem Benutzerkontext zu starten. Prinzipiell können nicht nur Webserver mit FastCGI-Applikationen kommunizieren, sondern jedes Programm, das Zugang zu dem Kommunikationsendpunkt hat. Bei TCP-Verbindungen können alle Programme auf Rechnern, die zum selben Port Verbindung aufbauen können, auch mit der Applikation interagieren. Bei lokalen Ressourcen, z.B. Sockets, können Applikationen von jedem Programm auf dem lokalen Rechner theoretisch diesen Punkt ansprechen. Eine zuverlässige Trennung ist aufgrund der technischen Realisierung von FastCGI nicht möglich. Mechanismen zur Beschränkung des Zugriffs sind nur in geringem Umfang möglich, z.B. könnte eine FastCGI-Applikation so konfiguriert werden, sodass nur Verbindungen von bestimmten IP-Adressen möglich sind. Das TCP sollte daher nur verwendet werden, wenn der Webserver und die Server im abgesicherten Netz genutzt werden und alle Computer im Netz vertrauenswürdig sind. Zusätzlich sollten keine Netzverbindungen von außen auf TCP-Port initiiert werden (z.B. durch Paketfilter). [Mün02] (S.104-105)

Im Rahmen der Recherche gibt es viele Möglichkeiten den eigenen Apache-Webserver und damit auch das CGI und FastCGI abzusichern bzw. sicherer zu gestalten. Pauschal lässt sich nicht sagen, ob CGI oder FastCGI sicherer ist. Es hängt von den Vorkehrungen ab, die der Programmierer bzw. Administrator bereit ist zu implementieren. In Bezug auf den Zugriff auf das Skript stehen aufgrund der technischen Bauweise von FastCGI nicht so viele Möglichkeiten zur Verfügung, um eine Sicherung einzubauen.





## 7. Fehleranalyse

Bei der Implementierung und dem Vergleich sind bestimmte Aspekte aufgefallen, die zu Ungenauigkeiten bei der Messung der Daten geführt haben können. Es kann sein, dass beim Messen mit dem Apache Benchmarking Tool Ungenauigkeiten aufgetreten sind. Es besteht die Möglichkeit noch weitere Benchmarking Tools zu verwenden, um einen größeren Vergleich zu schaffen und eine genauere Aussage zu treffen. Obwohl CGI und FastCGI nicht mehr weiterentwickelt werden, kann eine weitere Ursache für Messungenauigkeiten, die Version der Technologien selbst sein, da diese nicht aufeinander abgestimmt sein können. Eine weitere Quelle für Messfehler kann die fehlende Optimierung der CGI- und oder FastCGI-Skripte sein. Dies ist hauptsächlich auf die fehlende Erfahrung im Umgang mit den Technologien zurückzuführen. Es hätte mehr Zeit investiert werden können, um sich mit dem Thema noch tiefgreifender auseinanderzusetzen. Die meisten Dokumentationen oder Informationen aus dem Internet sind spärlich und veraltet. Sie widersprechen sich zum Teil auch gegenseitig, wodurch die Implementierung erschwert wird. Eine Investition in neue Bücher oder andere neuwertige Quellen sind möglich, aber nicht notwendig und eine Recherche in größeren Bibliotheken hätte diesen Prozess unterstützt. Die verwendete Hardware kann aus unbekannten Gründen für eine der beiden Technologien optimierter sein als für die andere. Zusätzlich könnte ein weiterer Computer mit identischer Hardware aufgebaut werden, um sicherzustellen, dass die Hardware nicht defekt ist. Die Performancetests haben die CPU kurz zu 100% ausgelastet. Dies hat sich auf die Temperatur der CPU ausgewirkt, da sie stellenweise bis zu 15°C wärmer geworden ist als im Leerlauf. Folglich kann sich dies auf die Performance auswirken, da die Tests direkt hintereinander durchgeführt wurden. Eine Lösung dafür wäre es, die CPU abkühlen zu lassen und/oder den Computer nach jedem Versuch neu zu starten. Der in diesem Projekt erstellte Anwendungsfall ist simpel gehalten. Da es keine zusätzlichen Datenbankabfragen gibt, kann davon ausgegangen werden, dass FastCGI kaum bis keine Verbesserung gegenüber CGI bringt. Eine komplexere Anwendung mit einer Datenbankabfrage würde eventuell ein deutliches Ergebnis bringen. XAMPP kann ebenfalls Auswirkungen auf die Tests haben, da CGI und FastCGI möglicherweise besser oder eingeschränkt mit dem Programmpaket laufen. Eine separate Installation von den Anwendungen aus XAMPP ist als Vergleich möglich. Nach jedem einzelnen Test wird XAMPP über den Windows Task-Manager beendet, um sicherzustellen, dass das Programm vollständig geschlossen ist. Es kann jedoch sein, dass im Hintergrund noch Prozesse vom vorherigen Test laufen, was wiederum negative Auswirkungen auf die nachfolgenden Tests hätte. Dahingehend würde ein Neustart des Computers helfen. Zusätzlich kann es noch andere Faktoren geben, die noch nicht betrachtet wurden.



## 8. Fazit und Ausblick

Dynamische Webseiten sind heutzutage unerlässlich, um komplexere Inhalte darzustellen. CGI und FastCGI sind zwei ältere Technologien für die Generierung dynamischer Webseiten, welche von JavaScript und Alternativen abgelöst wurden. Dieses Studienprojekt hat das Thema nur grob angeschnitten und es gibt noch eine Vielzahl an Möglichkeiten und Features von CGI und FastCGI, die hier nicht dargestellt bzw. genutzt wurden. Zusammenfassend lässt sich sagen, dass der Vergleich an sich gelungen ist, auch wenn es im praktischen Teil kaum einen Unterschied in Bezug auf die Performance gibt. Die Umsetzung der Skripte hat funktioniert, obwohl es einige Probleme gab, da es kaum aktuelle Dokumentationen gibt und Unsicherheiten, ob die Implementierung dahingehend korrekt durchgeführt wurde. Um den praktischen Vergleich aussagekräftiger zu gestalten, kann der interne Prozess um Datenbankabfragen erweitert werden oder die Logik komplizierter gestaltet beziehungsweise auch für mehr Funktionalitäten ausgebaut werden. Falls Interesse zum auszuprobieren besteht, gibt es ein GitHub-Repository, welches mit der folgenden URL zu erreichen ist: <https://github.com/Marvun/cgi-fastcgi>.





## 9. Literaturverzeichnis

- Apache Friends*. (kein Datum). Abgerufen am 29. 07 2022 von <https://www.apachefriends.org/de/index.html>
- Böhringer, J., Bühler, P., & Schlaich, P. (2003). *Projekte zur Mediengestaltung - Briefing Projektmanagement Making of ...* Berlin, Heidelberg: Springer.
- Böhringer, J., Bühler, P., Schlaich, P., & Sinner, D. (2014). *Kompendium der Mediengestaltung - IV. Medienproduktion Digital*. Berlin, Heidelberg: Springer Vieweg.
- Brown, M. R., & mcarbonneaux, g. v. (29. April 1996). *FastCGI Specification*. Abgerufen am 29. 06 2022 von [https://fastcgi-archives.github.io/FastCGI\\_Specification.html](https://fastcgi-archives.github.io/FastCGI_Specification.html)
- Bühler, P., Schlaich, P., & Sinner, D. (2021). *Digitalmedien-Projekte - Briefing · Planung · Produktion*. Wiesbaden: Springer Fachmedien.
- Coar, K., & Robinson, D. (2004). *The Common Gateway Interface (CGI) Version 1.1*. The Apache Software Foundation (Network Working Group). Von <https://datatracker.ietf.org/doc/html/rfc3875> abgerufen
- Münch, I. (2002). *Apache Webserver Sicherheitsstudie*. Bonn: Bundesamt für Sicherheit in der Informationstechnik.
- Perl*. (kein Datum). Abgerufen am 3. August 2022 von <https://www.perl.org/>
- php*. (kein Datum). Abgerufen am 15. Juli 2022 von <https://www.php.net/manual/de/intro-whatis.php>
- Rabe, L. (3. Dezember 2021). *Statista*. Abgerufen am 25. April 2022 von <https://de.statista.com/statistik/daten/studie/805920/umfrage/anzahl-der-internetnutzer-weltweit/>
- Rabe, L. (03. Juni 2022). *Statista*. Abgerufen am 12. Mai 2022 von <https://de.statista.com/statistik/daten/studie/3337/umfrage/die-populaersten-internetseiten-weltweit/>
- Rabe, L. (24. Januar 2022). *Statista*. Abgerufen am 12. April 2022 von <https://de.statista.com/themen/2033/internetnutzung-in-deutschland/>
- Rohr, M. (2018). *Sicherheit von Webanwendungen in der Praxis - Wie sich Unternehmen schützen können - Hintergründe Maßnahmen, Prüfverfahren und Prozesse*. Wiesbaden: Springer Vieweg.



Shipley, P. (27. Juni 2019). *Improve PHP performance with FastCGI on XAMPP for Windows*. Abgerufen am 14. Juli 2022 von <https://paulshipley.id.au/articles/coding-tips/improve-php-performance-with-fastcgi-on-xampp-for-windows/>

Statista Research Department. (05. August 2022). *Statista*. Abgerufen am 09. August 2022 von <https://de.statista.com/statistik/daten/studie/181588/umfrage/marktanteil-der-meistgenutzten-webserver/>

*VirtualBox*. (kein Datum). Abgerufen am 04. August 2022 von <https://www.virtualbox.org>



## 10. Eidesstaatliche Erklärung

Wir erklären ehrenwörtlich:

1. dass wir unser Studienprojekt selbständig verfasst haben,
2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben,
3. dass wir unser Studienprojekt bei keiner anderen Prüfung vorgelegt haben.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Luckenwalde, 16.08.2022

*Knothe*

---

Ort, Datum und Unterschrift

Marvin Knothe

Werder, 16.08.2022

---

Ort, Datum und Unterschrift

Yüxi Zhang