universität wien

DIPLOMARBEIT / DIPLOMA THESIS

Titel der Diplomarbeit / Title of the Diploma Thesis

„Vermittlung von Informatikkonzepten anhand der Zahlenrätsel von Ephesos“

verfasst von / submitted by

Diana Altmann

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Magistra der Naturwissenschaften (Mag.rer.nat.)

Wien, 2016 / Vienna, 2016

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 190 884 299
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Lehramtsstudium UF Informatik und Informatikmanagement und UF Psychologie und Philosophie
Betreut von / Supervisor:	ao. Univ.-Prof. i.R. Dr. Erich Neuwirth
Mitbetreut von / Co-Supervisor:	

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.
Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Wien, im Mai 2016

Diana Altmann

Kurzfassung

Meine Diplomarbeit hat die Zahlenrätsel von Ephesos zum Thema. Diese wurden sowohl theoretisch als auch praktisch erarbeitet.

Auf der theoretischen Ebene wurden vier Unterrichtsszenarien entworfen, welche eine Vermittlung von ausgewählten Informatikkonzepten anhand der Zahlenrätsel von Ephesos in der AHS-Oberstufe ermöglichen sollen. Diese Ausarbeitungen sind jeweils so aufgebaut, dass zuerst allgemeine Informationen gegeben werden, gefolgt von einer Planungsmatrix und Code-Beispielen. Die Theorie zu den jeweiligen Themen befindet sich in zwei extrigen Kapiteln („3. Technische Grundlagen“ und „6. Geschichtlicher Hintergrund“), welche u.a. auch nutzbar für einen interdisziplinären Unterricht (Informatik in Verbindung mit Geschichte) ist.

Auf der praktischen Ebene wurde ein Programm entwickelt, welches eine Entschlüsselung der Zahlenrätsel von Ephesos ermöglicht. Dieses Programm wird auch in der Forschung vom Institut für Alte Geschichte an der Universität Wien verwendet und es konnten damit auch bereits erste, u.a. auch medienwirksame, Resultate erzielt werden.

Schlagwörter: Ephesos, Zahlenrätsel von Ephesos, Inschriften, HTML, CSS, PHP, SQL, Kryptographie, Verschlüsselung, Unterrichtsszenarien

Abstract

This diploma thesis deals with the number puzzles of Ephesus from a theoretical as well as a practical perspective.

On the theoretical level, four classroom scenarios for the “AHS-Oberstufe” were created. The aim of these scenarios is to convey selected concepts of computer science on the basis of the number puzzles of Ephesus. For each of the four scenarios some general information is presented first, followed by a planning matrix and code examples. The theoretical background for each of the topics can be found in two separate chapters (“3. Technische Grundlagen” and “6. Geschichtlicher Hintergrund”) and is also applicable for interdisciplinary teaching (computer science and history).

On the practical level, a program was developed which allows to decode the number puzzles of Ephesus. This programme is also used for research at the Institute for Ancient History at the University of Vienna and could already provide some results, thereby also attracting media attention.

Keywords: Ephesus, the number puzzles of Ephesus, epigraphs, HTML, CSS, PHP, SQL, encryption, teaching scenarios

Inhaltsverzeichnis

1. Vorwort

In diesem Vorwort möchte ich den Leserinnen und Lesern einen Einblick in den Aufbau dieser Diplomarbeit ermöglichen.

Nach diesem Vorwort befindet sich in **Kapitel 2** die Einleitung.

Das **Kapitel 3** beschreibt die technischen Grundlagen und Technologien, welche in Bezug auf das Projekt „Entschlüsselung der Zahlenrätsel von Ephesos“ und damit ebenfalls bezüglich der Unterrichtsszenarien verwendet wurden. Themen der theoretischen Ausarbeitung waren die Programmiersprache PHP, Datenbanken, Webserver und Kryptographie.

In **Kapitel 4** wird der Fokus auf die Beschreibung der konkreten Umsetzung des interdisziplinären Projekts „Entschlüsselung der Zahlenrätsel von Ephesos“ gelegt. Zuerst wird erklärt, worum es sich bei den Zahlenrätseln von Ephesos handelt. Anschließend werden Umrechnungstabelle, Datenbank und Web Interface behandelt – wobei bei letzterem die Implementierung des Projekts in PHP, die gesamte Code Dokumentation und der Ablauf des Programms eine große Rolle spielen. Abgeschlossen wird dieses Kapitel mit den Ergebnissen, welche bereits mit Hilfe dieses Programms in der Forschung erzielt werden konnten.

Kapitel 5 beinhaltet vier konkrete Unterrichtsszenarien mit Code-Beispielen zum Thema „Vermittlung von Informatikkonzepten anhand der Zahlenrätsel von Ephesos“, welche zeigen sollen, wie man dieses Projekt beispielsweise in den Informatik-Unterricht an der AHS-Oberstufe miteinbeziehen könnte. Die Titel der vier Szenarios lauten wie folgt: „Grundlagen von Programmiersprachen anhand von PHP und dessen Einbindung in Webstrukturen“, „Fortgeschrittene Konzepte von Programmiersprachen anhand von PHP“, „Datenbankgrundlagen und Einbindung in PHP-Code“ und „Kryptographie: Theorie und Praxis“.

In **Kapitel 6** geht es um den geschichtlichen Hintergrund dieses Projekts. Darin wird über die antike Metropole Ephesos, die Hanghäuser von Ephesos (in welchen die Zahlenrätsel von Ephesos gefunden wurden) und über Inschriften allgemein berichtet – alles zentrale Aspekte in Hinblick auf die Zahlenrätsel von Ephesos.

Schlussendlich befindet sich in **Kapitel 7** eine Zusammenfassung über die Diplomarbeit sowie deren Ergebnisse.

Dankesworte möchte ich an dieser Stelle an meine beiden Betreuer – Herrn a.o. Univ.-Prof. i.R. Dr. Neuwirth (Informatik) und Herrn ao. Univ.-Prof. Dr. Taeuber (Alte Geschichte) – richten, die mich stets tatkräftig unterstützt haben. Weitere Dankesworte gelten all denjenigen, die mir ebenfalls immer mit Rat und Tat beiseite gestanden sind.

2. Einleitung

Ich möchte mit meiner Diplomarbeit einen Beitrag für eine lebhafte Vermittlung von Informatikkonzepten (anhand der Zahlenrätsel von Ephesos) leisten, da ich es sehr wichtig finde, Begeisterung im Unterrichtsfach Informatik zu vermitteln. Meine Ausarbeitung soll Vorschläge in diese Richtung geben, wie dies beispielsweise umgesetzt werden kann.

Im theoretischen Teil der Arbeit habe ich dafür zuerst Grundlagen bestimmter Informatikkonzepte zusammengefasst. So wurden die Programmiersprache PHP, Datenbanken, Webserver und Kryptographie genauer erörtert (auch in Hinblick auf die Verwendung im Unterricht). In den vier ausgearbeiteten Unterrichtsszenarien wurden diverse Informatikkonzepte mit Hilfe von Code-Beispielen verdeutlicht, die auch auf diese Weise im Unterricht verwendet werden können. Das „Zahlenrätsel von Ephesos"-Projekt dient dabei als Modell und Veranschaulichung – interdisziplinär betrachtet ist daher auch der geschichtliche Hintergrund interessant. Dieser wird in einem extrigen Kapitel behandelt und kann ebenfalls im Unterricht eingesetzt werden, insbesondere kann damit fächerübergreifend mit dem Unterrichtsfach Geschichte gearbeitet werden.

Der praktische Teil beinhaltet die Implementierung der Entschlüsselung der Zahlenrätsel von Ephesos, wobei hierbei besondere Aspekte zu berücksichtigen gab, wie z.B. diverse Ausnahmen und eine Entschlüsselung in beide Richtungen. Des Weiteren war es wichtig, nur griechische Namen berücksichtigen, die es zur damaligen Zeit gab. Dieses Programm wurde vom Institut für Alte Geschichte (Vorstand: ao. Univ.-Prof. Dr. Hans Taeuber) benötigt, da es diese besondere Form der Namens- und Zahlenberechnung, unter Berücksichtigung von Ausnahmen und dieser speziellen Menge an griechischen Namen, noch nicht gab, aber für Forschungstätigkeiten vonnöten war. Da sich dieses Projekt gut für die Veranschaulichung diverser Informatikkonzepte eignete, hat es sich deshalb angeboten, interdisziplinär (Informatik in Verbindung mit Geschichte) erarbeitet zu werden. Ergebnisse, die bereits mit Hilfe dieses Programms in der Forschung erzielt werden konnten, finden sich in Kapitel „4.6 Ergebnisse".

Methoden, die in Bezug auf diese Diplomarbeit verwendet wurden, waren Recherche (der Theorie), Ausarbeitung (der Unterrichtsszenarien) und die konkrete Implementierung (des Programms für die Entschlüsselung der Zahlenrätsel von Ephesos).

3. Technische Grundlagen – Verwendete Technologien

Vorbemerkung: Diese Arbeit soll als Arbeitsmaterial für den Informatikunterricht geeignet sein. Sie beschreibt daher elementare Begriffe und Konzepte der Informatik ausführlicher, als das in einem rein technischen Dokument notwendig wäre.

In diesem Kapitel geht es um die technischen Grundlagen und um die Technologien, die in Bezug auf das Projekt „Entschlüsselung der Zahlenrätsel von Ephesos" verwendet wurden. Dabei wird zuerst auf die Programmiersprache eingegangen, welche für die Programmierung benötigt wurde. Danach werden Aspekte in Bezug auf Datenbanken erörtert und in aller Kürze vorgestellt, was ein Webserver ist und wozu er verwendet wird. Das Kapitel wird schlussendlich mit einer Ausarbeitung zum Thema Kryptographie beendet.

3.1 PHP

3.1.1 Allgemeines

PHP ist also die Programmiersprache, die ich für die Umsetzung dieses Projekts gewählt habe. Warum ich mich gerade für diese entschieden habe, soll auf den nächsten Seiten klar werden. Ein großer Vorteil dieser Programmiersprache ist, dass sie schnell erlernbar ist und gleichzeitig imstande ist, sehr viel zu leisten, besonders in Bezug auf Webentwicklungen. Das Zusammenspiel von PHP und HTML ist auch lobenswert und unkompliziert, sodass der Übergang von einer statischen zu einer dynamisch generierten Webseite ohne Hindernisse möglich war.

Was zeichnet PHP als Programmiersprache aus? Wie bzw. auf welche Weise lässt es sich damit arbeiten? Diese und weitere Fragen sollen nun im Folgenden beantwortet werden.

Zuerst zur Abkürzung „PHP“, welche unter Umständen etwas verwirrend erscheint, da das erste Wort der Abkürzung bereits die Abkürzung ist (was auch rekursives Akyronym und Backronym genannt wird[1]) – denn so steht PHP für „**P**HP: **H**ypertext **P**reprocessor“ (ursprünglich für „**P**ersonal **H**ome **P**age Tools“). PHP ist eine weitverbreitete Open Source Programmier- und Script-Sprache, speziell für Webentwicklungen. Eines der Hauptziele dieser Programmiersprache ist, Webentwicklern die Möglichkeit zu geben, schnell dynamisch generierte Webseiten zu erzeugen.[2] Dem kann ich persönlich auch, wie bereits angesprochen, vollständig zustimmen.

PHP gehörte bis einschließlich Version 4.x zu den prozeduralen Programmiersprachen, ab Version 5.0 kann PHP auch zu den objektorientierten Sprachen gezählt werden. Kurz zur

[1] URL: https://de.wikipedia.org/wiki/PHP, aufgerufen am 6.11.2015.
[2] vgl. URL: http://php.net/manual/de/faq.general.php, aufgerufen am 2.10.2015.

Erklärung: Bei erstgenannten Programmiersprachen existiert eine Anweisungsreihenfolge, die abgearbeitet wird. Letztgenannte Sprachen beherrschen regulierbare Datenkapselung bei Objekten und Vererbung. PHP bietet beide Formen der Programmierung an.[3]

Wie sieht es mit dem PHP-Code und dessen Kompilierung aus? PHP-Code wird üblicherweise nicht in statische, für das Betriebssystem ausführbare Dateien übersetzt, sondern von einem Interpreter zur Laufzeit spontan kompiliert und ausgeführt (siehe „3.3 Webserver"), welches ein typisches Kennzeichen von Script-Sprachen ist. Für den Programmierer bzw. die Programmiererin bedeutet dies, dass für ihn bzw. sie jegliches Neukompilieren vor jedem neuen Testen entfällt. Des Weiteren muss er bzw. sie sich auch nicht selbst um die Arbeitsspeicherverwaltung kümmern, was ebenfalls typisch für Script-Sprachen ist.[4]

PHP ist mittlerweile auch (ähnlich wie Perl) als Script-Sprache auf Konsolenebene einsetzbar, doch der eigentliche und konsequent verfolgte Einsatzzweck ist nach wie vor der als Script-Umgebung eines Webservers (siehe „3.3 Webserver"). Wie bereits kurz angesprochen, kann PHP bequem in HTML eingebettet werden und PHP-Dateien können ebenso wie HTML-Daten in allen Webverzeichnissen abgelegt und ausgeführt werden. Ein weiterer großer Vorteil von PHP ist, dass es eine besondere Bindung an die Gegebenheiten von HTML und HTTP aufweist. So übernehmen beispielsweise viele vordefinierte Funktionen Aufgaben, die speziell der Webprogrammierung entgegenkommen.[5] Von diesen konnte ich auch selbst beim Programmieren profitieren – inwiefern wird in den nächsten Kapiteln gezeigt.

3.1.2 PHP – Von der Entwicklung bis zur heutigen Nutzung

Interessant in Bezug auf eine heutzutage sehr erfolgreiche Programmiersprache ist, wie ich finde, wie es zu derartigem Erfolg gekommen ist – denn wie jede Programmiersprache hat auch PHP einmal „klein" angefangen. Diesen Verlauf möchte ich im Folgenden kurz zusammenfassen.

Die Geschichte von PHP reicht bis in das Jahr 1995 zurück. Drei Jahre später gelang dann der Aufstieg von einem kleinen Toolset zu einer ernstzunehmenden Technologie auf Basis der Version 3.0. Damit wuchs aber zugleich auch der Druck auf Stabilität und Features der Script-Sprache.[6]

[3] vgl. Münz 2008, S. 639.
[4] vgl. Münz 2008, S. 639.
[5] vgl. Münz 2008, S. 639.
[6] vgl. Münz 2008, S. 639.

Für die Version 4.0 wurde der Script-Interpreter dann von Grund auf neu entwickelt – sein Kern war die sogenannte „Zend Engine“. Bereits in der langen Beta-Phase zog der neue Script-Interpreter immer mehr interessierte Webprogrammierer „in seinen Bann“. Nachdem PHP 4.0 im Mai 2000 offiziell freigegeben wurde, sorgte ungewöhnlich viel Werbung dafür, dass PHP in Bezug auf Webprogrammierung sehr bald zum Maß der Dinge wurde. Damit wurden auch zuvor dominierende Sprachen, wie z.B. Perl, aus diesem Bereich weitgehend verdrängt und auch Microsoft hat PHP bis heute keine Konkurrenz machen können, da das .NET-Framework den meisten Webprogrammierern zu komplex und zu proprietär ist. Andere Konkurrenten (wie z.B. die Java Server Pages) haben sich bestimmte Nischen gesichert, vor allem bei Banken und Großunternehmen. Dort wird Java auch für andere Zwecke verwendet und es haben Firmenchefs das Sagen, die beispielsweise einem kommerziellen Anbieter (Java wird ja von Sun Microsystems vertrieben) mehr vertrauen als einem Open-Source-Projekt wie PHP.[7]

Laut http://php.net/usage.php war PHP im Dezember 2004 auf 18,4 Millionen virtuellen Hosts (Domains) bzw. auf 1,3 Millionen echten Hosts (IP-Adressen) mit öffentlichen Webservern installiert. Das entspricht in etwa einem Drittel aller im öffentlichen Web erreichbaren Hosts.[8] Der Anteil wuchs seitdem (und heute immer noch) weiter rapide, wie an Abb. 1 ersichtlich:

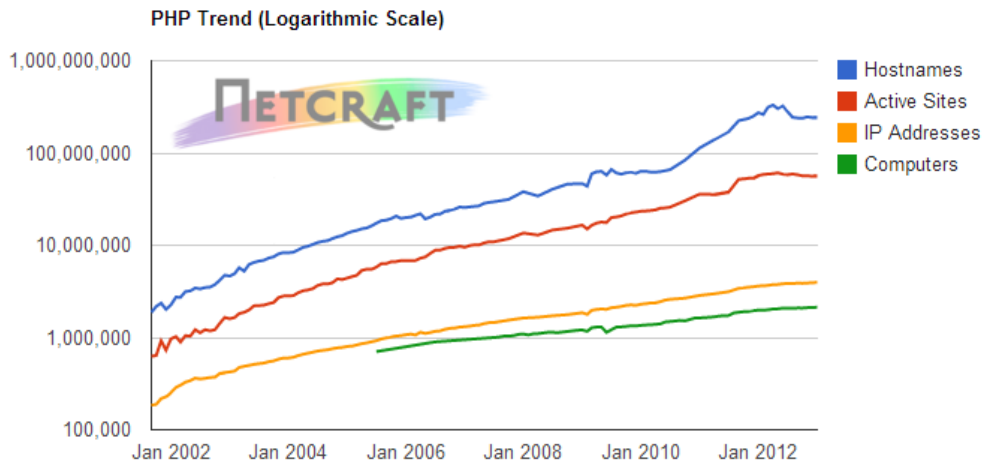


Abbildung 1 PHP 244 Mio. Seiten, 2,1 Mio. IP-Adressen laut Netcraft

Ab der Produktversion 5.0 im Jahr 2004 konnte PHP schließlich auch die Bedürfnisse von Großprojekten zufriedenstellen, die allein schon aus organisatorischen Gründen streng objektorientiert arbeiten.[9]

[7] vgl. Münz 2008, S. 640.
[8] vgl. Münz 2008, S. 640.
[9] vgl. Münz 2008, S. 640.

Als Nachfolger von PHP 5 war eigentlich PHP 6 vorgesehen, aber dann wurde die Entwicklung eingestellt. Diskussionen darüber, wie die nächste PHP-Version heißen soll, wurden gestartet und schlussendlich konnte sich die Versionsnummer 7 durchsetzen. PHP 7 wird voraussichtlich eine um bis zu 30% geringere Ausführungszeit als PHP 5 haben (da Hashtabellen neu implementiert wurden) und die fertige Version soll im November 2015 veröffentlicht werden – also genau zur Entstehungszeit dieser Diplomarbeit.[10]

3.1.3 Strukturen einer PHP-Seite

In diesem Unterkapitel und in den darauffolgenden sollen grundlegende Strukturen und Schlüsselbegriffe, die in Bezug auf die Programmiersprache PHP und für das Projekt erforderlich waren, anhand von Code-Beispielen leicht verständlich erörtert werden.

Ausgangspunkt einer jeden PHP-Webapplikation ist eine einfache Textdatei mit der Endung „`.php`“. Eine gewisse Struktur (die eventuell bereits von HTML bekannt ist) sollte eingehalten werden. Weiters müssen PHP-Befehle in gesonderten Bereichen von anderweitigem Inhalt, wie z.B. HTML, getrennt sein. Ersichtlich ist dies in folgendem kurzen Code-Abschnitt:

```
<html>
	<head>
	</head>
	<body>
		<?php
			echo „<h1>Hallo Welt!</h1>“;
		?>
	</body>
</html>
```

In diesem Code-Abschnitt ist der PHP-Bereich fett hervorgehoben inmitten des HTML-Codes. Ab der Zeile mit `<?php` (oder kürzer: `<?`) werden alle Anweisungen vom PHP-Parser abgearbeitet, bis der Bereich mit einem `?>` endet. Code außerhalb des PHP Bereichs wird vom PHP-Parser überlesen. Mit dem Befehl `echo` erzeugt man eine Ausgabe am Bildschirm – hier dementsprechend „Hallo Welt!“. Nachdem der Parser das gesamte Skript verarbeitet hat, enthält es nur noch HTML.[11]

[10] vgl. URL: https://de.wikipedia.org/wiki/PHP, aufgerufen am 6.11.2015.
[11] vgl. Reimers und Thies 2012, S. 72-73.

Im Folgenden ist nun die Ausgabe des vorigen Skripts nach dem Parsen ersichtlich:

```
<html>
	<head>
	</head>
	<body>
		<h1>Hallo Welt!</h1>
	</body>
</html>
```

Man kann erkennen, dass die Definition des HTML- (`html`), Kopf- (`head`) und Körperbereichs (`body`) vom Parser nicht beachtet wurde und der Befehl `echo` zu einer Überschrift erster Ordnung (`h1`) geworden ist.

3.1.4 Variablen

Variablen sind dazu da, um Daten während der Abarbeitung eines Skriptes zu speichern. Wenn man eine Variable deklariert und ihr einen Wert zuweist, ist das nichts Anderes, als dem Wert einen Namen zu geben. Benötigt man diesen Wert zu einem späteren Zeitpunkt im Skript wieder, so kann man ihn über den vergebenen Namen ansprechen. Der Wert muss allerdings keineswegs immer derselbe sein – denn so ist es möglich, einer bestimmten Variablen einen neuen Wert zuzuweisen (welche den bisherigen dann ersetzt). Die Bindung eines Namens an einen Wert wird über den Zuweisungsoperator („=“) vorgenommen. Eine gültige Zuweisung sieht dann in Pseudocodeform z.B. wie folgt aus: „`Name = Wert;`“.[12]

Grundlegende Syntax

Jeder Variablenwert ist in einem PHP-Code bzw. in einem Gültigkeitsbereich (engl. „scope“) eindeutig. Gültige Namen werden immer durch ein Dollarzeichen („$“) eingeleitet, zwingend gefolgt von einem Buchstabe oder einem Unterstrich („_“) – danach kann der Name eine beliebige Anzahl von Buchstaben, Zahlen und Unterstrichen enthalten, wie in folgendem Code ersichtlich:[13]

```
<?php
	$testvariable1 = 'Text';
	$_testvariable2 = '123';
	$3tevariable = 1.5;
?>
```

Die beiden ersten festgelegten Variablen (`$testvariable1` und `$_testvariable2`) sind gültige Variablennamen; die letzte hingegen (`$3tevariable`) ist kein gültiger Variablenname. Wenn eine Variable einmal belegt ist, lässt sich ihr Inhalt so lange über den vergebenen Name ansprechen, bis sie mit dem Befehl `unset()` gelöscht wird oder das Skript beendet wird. Ob

[12] vgl. Reimers und Thies 2012, S. 74.
[13] vgl. Reimers und Thies 2012, S. 74.

eine Variable existiert, kann man wiederum über den Befehl `isset()` herausfinden (welcher `true` oder `false` zurückliefert, je nachdem ob die Variable im aktuellen Gültigkeitsbereich definiert ist). Wichtig ist, dass PHP bei Variablennamen zwischen Groß- und Kleinschreibung unterscheidet.[14] Deshalb sind die folgenden beiden Variablen auch nicht identisch:

```
<?php
      $variable = 1; // $variable wird der Wert 1 zugewiesen
      $Variable = 2; // $Variable wird der Wert 2 zugewiesen
      $variable = $Variable; // diese Zuweisung verändert den Wert
      // der Variable $variable von 1 auf 2
?>
```

Man sieht, dass die Variablen `$variable` und `$Variable` voneinander vollständig unabhängig sind. Nach der Belegung von `$Variable` enthält die Variable `$variable` immer noch den Wert 1. Erst durch die Zuweisung am Ende wird der Wert von `$variable` mit dem Wert von `$Variable` überschrieben und beträgt danach 2.[15]

Datentypen

Wenn man Datentypen für Werte definiert, legt man gleichzeitig die Operationen fest, mit denen man die Daten bearbeiten kann. So lässt sich beispielsweise die Zeichenkette „Das ist ein Satz“ schwer zu einer Zahl wie z.B. 5 addieren, da es ja nicht möglich ist, mit Sätzen zu rechnen. Des Weiteren ist durch den Datentyp festgelegt, welche Werte für die Operationen gültig sind. In PHP stehen insgesamt acht Datentypen zur Verfügung, wie an folgender Tabelle ersichtlich (Abb. 2).[16]

Bezeichner	Datentyp	Beispiel/Beschreibung
Zeichenketten	String	`'a', 'Wort', 'ganze Sätze'`
Ganzzahlige Werte	Integer	`1, 2, 3, 100, 1000, 1, 15`
Fließkommazahlen	Float/Double	`1.5, 11.99999, 17.4e2`
boolesche Werte	Boolean	`true, false`
Arrays	Array	mehrwertiger Datentyp
Objekte	Object	mehrwertiger Datentyp
Ressourcen	Resource	Referenz auf externe Quellen
Null	Null	Typ für Variablen ohne Wert

Abbildung 2 PHP-Datentypen

Datentypen können per Befehl abgefragt und verändert werden – für die meisten Datentypen sind dafür sogar eigene Funktionen vordefiniert (siehe Abb. 3):

[14] vgl. Reimers und Thies 2012, S. 74.
[15] vgl. Reimers und Thies 2012, S. 74-75.
[16] vgl. Reimers und Thies 2012, S. 76.

Datentyp	Konvertierung	Abfrage
String	(string)	is_string
Integer	(int), (integer)	is_int, is_integer
Float/Double	(float), (double)	is_float, is_double
Bool	(bool), (boolean)	is_bool
Array	(array)	is_array
Object	(object)	is_object
Null	-	is_null

Abbildung 3 Funktionen zur Typprüfung und -konvertierung

Nun zu wichtigen Begriffen in Bezug auf Datentypen in PHP:

Zeichenketten: Zeichenketten (auch „Strings“ genannt) stehen immer in einfachen (’) oder doppelten Anführungszeichen (“) und können alle möglichen Zeichen enthalten – also nicht nur Buchstaben, sondern auch Zahlen und Sonderzeichen. Der Ausdruck `’123’` ist ein String, auch wenn er nur aus Ziffern besteht und das nur deshalb, weil er in Hochkommas eingeschlossen ist. Wichtig ist, dass man Anführungszeichen konsistent verwendet. Wenn man also z.B. eine Zeichenkette mit einem doppelten Anführungszeichen beginnt, muss man sie auch mit einem doppelten enden lassen.[17]

Da das Arbeiten mit Anführungszeichen oftmals problematisch verläuft und es bestimmte Aspekte zu berücksichtigen gibt, sollen die beiden Arten (einfache und doppelte Anführungszeichen) hier kurz genauer erklärt werden:

Einfache Anführungszeichen: Eine Zeichenkette, die in einfachen Anführungszeichen eingeschlossen ist, wird fast ohne Verarbeitung ausgegeben, was bedeutet, dass darin enthaltene Variablen oder Sonderzeichen nicht ausgewertet werden.[18]

Ein Beispiel lautet wie folgt:

```
$var = ’Test’;
echo ’Ausgabe einer $var-Variablen’;
```

Man könnte annehmen, dass hier als Ausgabe „Ausgabe einer Test-Variablen“ erscheint, aber stattdessen ist „Ausgabe einer `$var`-Variablen“ zu lesen.

Doppelte Anführungszeichen: Strings in doppelten Anführungszeichen werden vom PHP-Parser bei ihrer Verwendung verarbeitet. Das bedeutet, dass sowohl alle darin enthaltenen Zeichen sowie Sonderzeichen als auch die Variablen aufgelöst werden – und dabei in den Datentyp „String“ konvertiert werden.[19] Das vorherige Beispiel nun mit doppelten Anführungszeichen führt zur wohl naheliegenden Ausgabe „Ausgabe einer Test-Variablen“:

```
$var = “Test“;
echo “Ausgabe einer $var-Variablen“;
```

[17] vgl. Reimers und Thies 2012, S. 78.
[18] vgl. Reimers und Thies 2012, S. 78.
[19] vgl. Reimers und Thies 2012, S. 79.

Ein weiterer Aspekt, mit dem es sich, in Bezug auf Anführungszeichen, zu befassen lohnt, und manchmal auch unumgänglich ist, ist die sogenannte Maskierung. Was man darunter versteht, soll in den folgenden Zeilen erörtert werden:

Maskierung: Strings mit doppelten Anführungszeichen werden vom PHP-Parser bei ihrer Verwendung verarbeitet und es werden alle darin enthaltenen Zeichen sowie Sonderzeichen als auch die Variablen aufgelöst (und dabei in den Datentyp „String“ umgewandelt). Des Weiteren werden Sonderzeichen, wie beispielsweise der Zeilenumbruch, richtig interpretiert. Andere Sonderzeichen, wie z.B. der Backslash („\“) oder das Dollarzeichen („$“), müssen erst mit einem Backslash „maskiert“ werden (da diese ansonsten interpretiert werden).[20] Wie das Maskieren funktioniert, ist an folgendem Code ersichtlich:

```
echo “PHP-Variablen beginnen mit einem \$-Zeichen“;
```
[21]

Damit erreicht man die Ausgabe des Satzes „PHP-Variablen beginnen mit einem $-Zeichen“. Ohne Backslash vor dem Dollarzeichen würde der PHP-Parser versuchen, die folgenden Zeichen so weit wie möglich als Variablennamen auszuwerten. Anführungszeichen des jeweils anderen Typs werden in Zeichenketten vom PHP-Parser nicht als störend angesehen – ohne eine Form der Maskierung kann man also Strings wie z.B. `echo “’escapen’ bedeutet Zeichen maskieren“;` und `’“escapen“ bedeutet Zeichen maskieren’;` bilden und ausgeben lassen. Anders ist dies jedoch bei gleichen Anführungszeichen. In diesem Fall muss man die auszugebenden Zeichen maskieren, wie dies bei folgendem Code gemacht wurde:

```
echo “\“escapen\“ bedeutet Zeichen maskieren“
```
[22]

Da nun hoffentlich etwas Klarheit bezüglich Anführungszeichen und Maskierung geschaffen werden konnte, widmen sich die folgenden zwei Seiten weiteren wichtigen Datentypen, nämlich den ganzzahligen Werten, Fließkommazahlen und schließlich den boolschen Wahrheitswerten.

Ganzzahlige Werte: Sie haben keine Nachkommastellen und zu ihnen gehören die positiven und negativen Zahlen sowie die Null. Als Synonym für Ganzzahl ist auch der Begriff „Integer“ geläufig. Für Zahlen sind in PHP die vier Grundrechnungsarten definiert – also Addition (`+`), Subtraktion (`-`), Multiplikation (`*`) und Division (`/`). Zusätzlich gibt es noch den Modulo-Operator (`%`). Der Modulo zweier Zahlen ist der Rest, der übrig bleibt, wenn die erste Zahl durch die zweite geteilt wird – so hat z.B. die Modulorechnung `31%6` das Ergebnis 1. Ganzzahlige Werte werden häufig als Zählvariablen benutzt, um die Häufigkeit eines bestimmten Ereignisses herauszufinden. Die Variable wird dann z.B. mit 0 initialisiert und

[20] vgl. Reimers und Thies 2012, S. 79.
[21] Reimers und Thies 2012, S. 79.
[22] vgl. Reimers und Thies 2012, S. 79.

dann bei jedem Auftreten des Ereignisses um 1 hochgezählt („inkrementiert“). Eine Variable herunterzuzählen nennt man dementsprechend „dekrementieren“. Für beide Varianten gibt es in PHP Operatoren.[23] Um eine Variable mit dem Namen `i` zu erstellen, auszugeben und zu erhöhen, schreibt man Folgendes:

```
$i = 0;
echo $i++;
```

Als Ausgabe erhält man den Wert 0. Dies scheint auf dem ersten Blick eventuell verwunderlich, da man sich als Ausgabe unter Umständen 1 erwartet hätte. Wichtig in diesem Zusammenhang ist, dass `$i` erst *nach* dem echo-Befehl den Wert 1 hat. Für eine sofortige Inkrementierung und Ausgabe müsste man nämlich `echo ++$i;` schreiben. Die Methode mit dem nachgestellten Operator wird Post-Inkrementierung bzw. Post-Dekrementierung genannt, weil die Variable erst nach Ausführung des echo-Befehls erhöht bzw. verringert wird. Das Gegenteil dazu ist Prä-Inkrementierung bzw. Prä-Dekrementierung. Letztgenannte sieht wie folgt aus:[24]

```
$i = 1;
echo --$i;
```

Hier wird die Variable `$i` zuerst verändert, bevor das `echo` ausgeführt wird. Man erhält also dementsprechend die Ausgabe 0.

Fließkommazahlen: Zahlen mit Nachkommastellen werden Fließkommazahlen genannt. Die Programmierung kennt zwei verschiedene Arten, die in älteren Programmiersprachen, nicht jedoch von PHP, unterschieden werden: „Float“ und „Double“. Mit Fließkommazahlen können alle bereits angesprochenen arithmetischen Funktionen ausgeführt werden, zusätzlich hat man – im Gegensatz zu den Integers – die Kontrolle über die Anzahl der verwendeten Nachkommastellen. Die Ergebnisse diverser Berechnungen werden mit einer ganzen Reihe von Stellen hinter dem Komma angegeben. Diese Genauigkeit kann für ein weiteres Rechnen wohl wünschenswert sein, genauso gut kann aber auch auf eine bestimmte benötigte Anzahl von Nachkommastellen gerundet werden, wenn z.B. für eine Ausgabe nur eine begrenzte Genauigkeit benötigt wird.[25]

Boolesche Wahrheitswerte: Bei dieser Art von Wahrheitswerten gibt es nur zwei unterschiedliche Ausprägungen – nämlich wahr und falsch. In PHP werden sie mit den Begriffen `true` und `false` gekennzeichnet. Anwendung finden sie bei der Überprüfung zweier oder mehrerer Werte. Wenn man z.B. zwei Variablen auf Gleichheit prüfen will, muss man ein doppeltes Gleichheitszeichen („==“) als Vergleichsoperator verwenden, da das

[23] vgl. Reimers und Thies 2012, S. 84.
[24] vgl. Reimers und Thies 2012, S. 84.
[25] vgl. Reimers und Thies 2012, S. 85.

einfache Gleichheitszeichen ja bereits für die Zuweisung reserviert ist. So gibt beispielsweise `Wert1 == Wert2;` dann `true` oder `false` zurück; bei `Wert1 = Wert2;` hingegen überschreibt der `Wert2` den Wert von `Wert1`. Einschränkend muss aber angemerkt werden, dass die Prüfung auf Gleichheit zweier Werte mit dem doppelten Gleichheitszeichen nicht typsicher ist – eine typsichere Variante ist nur mit dem dreifachen Gleichheitszeichen („===“) möglich, was man an folgenden Beispielen erkennen kann: Die Abfrage `10 == '10';` ist `true`, da eine automatische Typkonvertierung vorgenommen wird. Der Ausdruck `10 === '10'` hingegen ergibt den Wert `false`, da eine Zahl mit einem String verglichen wird.[26]

3.1.5 Kommentare

Um selbst nicht die Orientierung im eigenen Programmcode zu verlieren, ist es wichtig, diesen zu kommentieren. Kommentare erleichtern es aber nicht nur dem Urheber des Codes selbst, den Überblick zu behalten, welche Funktion was bewerkstelligt, welche Codezeile was macht usw., sondern bzw. vor allem anderen Leuten, die sich mit dem Code ebenfalls beschäftigen (müssen). Wie diese Kommentare in PHP (bzw. HTML) aussehen, soll deshalb nun im Folgenden erörtert werden.

Kommentare sind also Zeilen oder Texte, die man mitten in den Code hineinschreiben kann und welche beschreiben sollen, was an dieser Stelle im Code gerade passiert. Ein angemessen kommentierter Code eignet sich wesentlich besser zur Wiederverwendung. Es gibt mehrere Arten, wie man Kommentare im Code einleiten kann:[27]

Für kurze Bemerkungen eignet sich ein einzeiliger Kommentar, welcher in PHP mit einem doppelten Slash („`//`“) eingeleitet wird und bis zum Zeilenende gilt. Einzeilige Kommentare müssen nicht beendet werden. Alle Anweisungen innerhalb der Kommentarzeile, die nach dem `//` stehen, werden vom Parser ignoriert. Ein einzeiliger Kommentar könnte wie folgt aussehen:[28]

```
// Dies ist ein einzeiliger Kommentar.
```

Erklärungen, die mehr Platz beanspruchen, kann man in mehrzeilige Kommentare schreiben. Sie beginnen mit `/*` und enden mit `*/`. Es gibt keine maximale Länge. Ein Beispiel für einen mehrzeiligen Kommentar sieht wie folgt aus:[29]

```
/* Dies ist ein mehrzeiliger Kommentar, welcher ebenfalls vom Parser
ignoriert wird und der z.B. dann angebracht ist, wenn man die
Funktion des Skripts allgemein beschreiben oder schwierig zu
verstehende Passagen erklären möchte. */
```

[26] vgl. Reimers und Thies 2012, S. 87.
[27] vgl. Reimers und Thies 2012, S. 100.
[28] vgl. Reimers und Thies 2012, S. 101.
[29] vgl. Reimers und Thies 2012, S. 101.

Da beim Programmieren mit PHP immer wieder auch Bereiche mit HTML vorkommen, soll an dieser Stelle auch noch kurz erklärt werden, wie man Kommentare in HTML einfügt. Ein Kommentar wird durch die Zeichenfolge `<!--` eingeleitet, dahinter folgt dann ein beliebig langer Kommentartext, bis der Kommentar von der Zeichenfolge `-->` schließlich beendet wird. Innerhalb des Kommentartextes können auch HTML-Elemente notiert werden, da alles, was zwischen der einleitenden Zeichenfolge und der beendenden Zeichenfolge steht, bei der Anzeige im Browser unterdrückt wird.[30] Ein Kommentar in HTML könnte also beispielsweise so aussehen:

```
<!-- Dies ist ein Kommentar in HTML. -->
```

3.1.6 Funktionen

Funktionen spielen bei Programmierung allgemein eine große Rolle – so auch in Bezug auf mein Projekt. Doch was genau kann man sich darunter vorstellen, wie sind sie aufgebaut und wozu kann man sie verwenden? Diese Fragen und weitere Aspekte sollen im Folgenden so gut wie möglich beantwortet werden.

Zuallererst ist zu sagen, dass Funktionen dazu verwendet werden, um bestimmte Probleme zu lösen. In einer Funktion werden Befehle gekapselt, die eine vorgegebene Aufgabe lösen – diese können simpel sein oder auch komplex. Funktionen werden über formale Definitionen eindeutig beschrieben: Dazu gehören die Angabe aller Eingabeparameter mit Name, Reihenfolge, Datentyp und Datentypinformationen (sofern es einen Datentyp gibt). Eingabeparameter werden der Funktion beim Aufruf zur Verarbeitung übergeben. Wird keinerlei Eingabeparameter benötigt, so spricht man von einer parameterlosen Funktion. Alle wichtigen Daten sind in diesem Fall in der Funktion festgeschrieben oder werden auf anderem Weg in die Funktion geholt. Nachdem der Rumpf der Funktion vollständig abgearbeitet ist, wird unter Umständen ein Rückgabewert geliefert – dieser kann z.B. eine Zahl sein oder ein boolescher Wahrheitswert, um über Erfolg oder Misserfolg der Ausführung zu berichten.[31]

Kapselung

Wie der Rumpf der Funktion implementiert ist, welche Programmierschritte also ausgeführt werden, ist für den Anwender bzw. die Anwenderin nicht wichtig zu wissen – sofern er bzw. sie die Definition kennt. Er bzw. sie sieht die Funktion also sozusagen als Blackbox. Dabei ist es nur wichtig zu wissen, *dass* und nicht *wie* die Funktion arbeitet. Das Verbergen von Implementierungsdetails wird Kapselung genannt und hat diverse Vorteile, welche nun genannt werden: Funktionen müssen nicht an dem Ort definiert werden, an der sie das erste

[30] vgl. URL: https://wiki.selfhtml.org/wiki/HTML/Regeln/Kommentar, aufgerufen am 6.11.2015.
[31] vgl. Reimers und Thies 2012, S. 102-103.

Mal aufgerufen werden, d.h. man kann sämtliche Funktionen an den Anfang oder das Ende des Codes verschieben (oder sogar in eine separate Datei auslagern) und erhält damit einen deutlicheren Überblick über die eigentliche Ablauflogik des Skripts. Ein weiterer Vorteil ist, dass man dieselbe Funktion mehrfach in unterschiedlichen Situationen nutzen kann (sofern man generalisierte Funktionen einsetzt, die sich also nicht auf einen Spezialfall beziehen). Die Wiederverwendung von Funktionen verkürzt effektiv das Skript, da kein Code unnötig dupliziert wird. Die vereinfachte Wartung des Codes sei noch als letzter positiver Aspekt der Kapselung angesprochen. Wenn man einen Fehler bei der Ausführung entdeckt, braucht man idealerweise nur einzelne Codezeilen einer oft eingesetzten Funktion korrigieren. Die Änderungen wirken sich sofort und auf alle weiteren Vorkommen aus.[32]

Syntax

Eingeleitet wird die Definition einer Funktion in PHP durch das Schlüsselwort `function`. Jede Funktion innerhalb eines Codes hat einen eindeutigen Namen. Die Eingabeparameter werden genau wie in der formalen Definition in runden Klammern angegeben, jeweils durch Kommas getrennt. Als Parameter gelten die Werte sowohl von Variablen als auch von Konstanten. Im Gegensatz zu anderen Programmiersprachen entfällt bei PHP die Angabe der Datentypen für die Parameter. Der Rumpf – also der gesamte Inhalt der Funktion – wird in geschweiften Klammern eingefasst, wie an folgendem Code ersichtlich:[33]

```
function multipliziere($parameter1,$parameter2)
{
      return $parameter1*parameter2;
}
```

Die Namen der Eingabeparameter sind frei wählbar und die Variablen sind lokal gebunden und damit nur innerhalb dieses Funktionsrumpfes nutzbar. Wird die Funktion mit `multipliziere(5,10);` aufgerufen, so erhält `$parameter1` den Wert 5 und `$parameter2` den Wert 10 zugewiesen und als Ergebnis der Funktion wird in diesem Fall 50 mittels `return` zurückgeliefert. Hierbei müssen zwei Parameter übergeben werden – werden zu viele übergeben, so werden die überzähligen einfach nicht verwendet. Die Angabe von zu wenigen Parametern führt allerdings zu einer Warnung. Parameter können auch als optional angelegt werden, indem man ihnen bereits bei der Funktionsdefinition einen sogenannten Default-Wert zuweist:[34]

[32] vgl. Reimers und Thies 2012, S. 103-104.
[33] vgl. Reimers und Thies 2012, S. 104.
[34] vgl. Reimers und Thies 2012, S. 105.

```
function quadriere($parameter = 2)
{
	return $parameter*$parameter;
}
```

Hier kann man also entweder einen Parameter angeben oder ihn weglassen. Gibt man einen an, dann wird mit diesem Wert gerechnet – ruft man die Funktion ohne Parameter auf, so bekommt `$parameter` den Wert 2. In diesem Beispiel beträgt der Rückgabewert 4.

3.1.7 Kontrollstrukturen

Da das Wort „Kontrollstrukturen" unter Umständen nicht allzu geläufig ist, soll hier zu Beginn dieses Kapitels einmal erklärt werden, was darunter im Kontext von Programmierung verstanden wird. Man benötigt diese, da Skripts grundsätzlich nicht einfach geradlinig verlaufen. So müssen z.B. in bestimmten Situationen Entscheidungen gefällt werden, deren Ausgang die weiteren Aktionen beeinflusst. Hier kommen dann die sogenannten Kontrollstrukturen ins Spiel, welche allgemein in drei Gruppen geteilt werden können: in bedingte Entscheidungen, Wiederholungen und Sprunganweisungen.[35] Im Folgenden werden jetzt jedoch nur die beiden ersteren genauer erörtert, da nur diese für die Programmierung des Projekts und für die Unterrichtsszenarien benötigt wurden.

Bedingte Entscheidungen

Bedingte Entscheidungen unterscheiden sich in der Anzahl der Handlungsalternativen, bei welchen es eine oder mehrere zu wählen gilt. Die Auswahl wird anhand von Kriterien getroffen: Im einfachsten Fall anhand eines einzigen Kriteriums – bei mehreren Kriterien werden diese im booleschen Sinn logisch zu einer Bedingung verknüpft und auf ihren Wahrheitsgehalt hin überprüft. PHP bietet zwei verschiedene Konstrukte an – das eine (`if`) eignet sich für komplexe Bedingungen, das andere (`switch`) für eine beliebig große Anzahl an Alternativen.[36] Nun werden beide erklärt und einander gegenübergestellt.

Das `if`-Konstrukt: Die einfachste Art einer Entscheidung hat nur genau eine Konsequenz. Man trifft hierbei also nicht die Entscheidung, *welche* Anweisungen ausgeführt werden, sondern nur, *ob* sie ausgeführt werden.[37]

Wenn `C` eintritt, dann `X` (siehe Abb. 4):

```
<?php
if($C)
{
	// Anweisungen der Konsequenz X ausführen
}
```[38]

[35] vgl. Reimers und Thies 2012, S. 110.
[36] vgl. Reimers und Thies 2012, S. 111.
[37] vgl. Reimers und Thies 2012, S. 111.

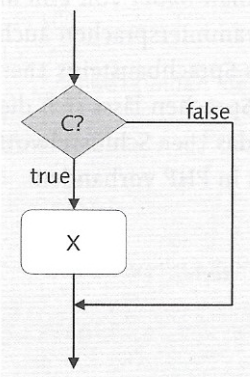


Abbildung 4 Struktur eines if-Konstrukts

`C` ist die zu prüfende Bedingung und die Konsequenz `X` steht für eine oder mehrere zusammengehörige Befehle. Wird die Bedingung `C` im booleschen Sinn zu `false` ausgewertet, wird `X` einfach übersprungen, bei `true` wird sie einmal vollständig ausgeführt. In den meisten Fällen gibt es aber mindestens zwei Handlungsalternativen, von denen immer nur eine eintreten kann. Bei genau zwei Alternativen lässt sich das `if`-Konstrukt wie folgt beschreiben:

Wenn `C` eintritt, dann `X`, anderenfalls `Y` (siehe Abb. 5).[39]

```
<?php
if($C)
{
	// Anweisungen der Alternative X ausführen
}
else
{
	// Anweisungen der Alternative Y ausführen
}
```
[40]

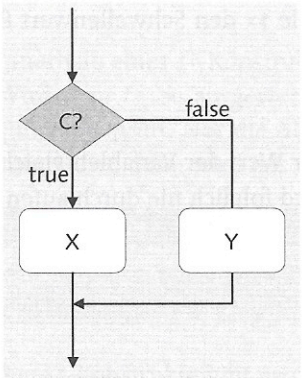


Abbildung 5 Auswahl zwischen zwei Handlungsalternativen

Bei mehreren Handlungsalternativen wird immer genau eine davon ausgeführt – im Beispiel ist dies die Alternative `X`, insofern `C` wahr ist, anderenfalls `Y`. Eine Handlungsalternative kann

[38] Reimers und Thies 2012, S. 111.
[39] vgl. Reimers und Thies 2012, S. 112.
[40] Reimers und Thies 2012, S. 112.

wiederum ein `if`-Konstrukt sein, usw. Wie man sieht, ist eine Schachtelung möglich, sodass sich auch Entscheidungen mit mehr als zwei Handlungsalternativen abbilden lassen, siehe Abb. 6.

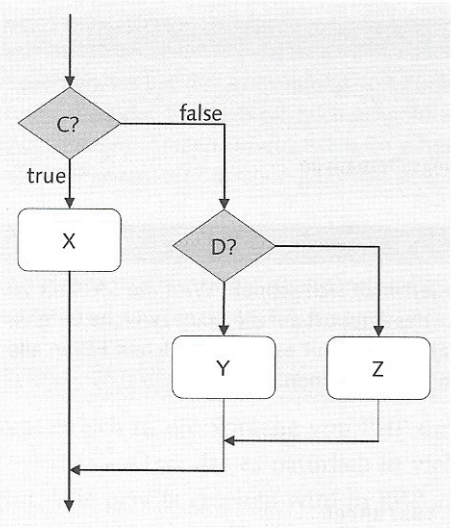


Abbildung 6 Verschachteltes if-Konstrukt

Der dazugehörige Code lautet wie folgt:

```
<?php
if($C)
{
	// Anweisungen der Alternative X ausführen
}
else
{
	if($D)
	{
		// Anweisungen der Alternative Y ausführen
	}
	else
	{
		// Anweisungen der Alternative Z ausführen
	}
}
```
[41]

Die Bedingungen `C` und `D` stehen in einer logischen Beziehung – wenn es zu einer Auswertung von `D` kommt, wurde `C` bereits als `false` erkannt. Bei der Definition von `D` ist es also wichtig, das Ergebnis von `C` im Hinterkopf zu behalten; anderenfalls kann es unter Umständen dazu kommen, dass mindestens eine der beiden Handlungsalternativen von `D` logisch unmöglich ist und ein `else`-Zweig nie durchlaufen wird.[42]

Das `switch`-Konstrukt: Dieses Konstrukt ist ausschließlich dafür geeignet, einen Wert mit einer vorher definierten Menge von Alternativen zu vergleichen. Anders als das `if` ist das

[41] Reimers und Thies 2012, S. 113.
[42] vgl. Reimers und Thies 2012, S. 113-114.

`switch` auf den exakten Vergleich (mittels „==“) begrenzt und es lassen sich mit einem `switch` auch nur einfache Typen wie Zeichenketten oder Zahlen prüfen – die Kontrolle von Arrays und Objekten ist somit nicht möglich. Das `switch`-Konstrukt besteht aus verschiedenen Teilen: Der Referenzwert, den es mit den Alternativen zu vergleichen gilt, wird durch das Schlüsselwort `switch` spezifiziert und jede Alternative wird innerhalb eines `case`-Blocks geprüft. In einem Block können beliebig viele PHP-Anweisungen stehen, die nur dann ausgeführt werden, wenn die Gleichheit zwischen Alternative und Referenzwert gegeben ist.[43]

```
switch($referenzwert)
{
      case ’100’:
      echo ’Der Referenzwert ist 100.’;
      break;
      case ’10’:
      echo ’Der Referenzwert ist 10.’;
      break;
}
```
[44]

Beim Ausführen des Codes werden die Alternativen der Reihe nach durchlaufen. Ist der Referenzwert gleich einer Alternative, welche durch ein `case` definiert ist, so werden nachfolgende Anweisungen ausgeführt. Jeder `case`-Block kann durch ein `break;` beendet werden, das dafür sorgt, dass das `switch`-Konstrukt verlassen wird. Somit wird nur die erste zutreffende Alternative ausgewertet – denn ohne den expliziten Abbruch würden alle nachfolgenden `case`-Blöcke ebenfalls durchlaufen und ausgeführt werden. Jedoch ist zu beachten, dass die Überprüfung auf Gleichheit nicht die Datentypkontrolle einschließt – so werden nämlich Referenzwert und Alternative über == und nicht über === verglichen. Da es vorkommen kann, dass keine der Alternativen ausgeführt werden (z.B. wenn nicht alle möglichen Fälle abgedeckt sind), so erscheint es sinnvoll, eine Anweisung standardmäßig auszuführen. Dafür existiert die Alternative `default`. Diese wird auf jeden Fall ausgeführt und daher ist sie immer die letzte, die es zu definieren gilt. Hierbei benötigt man nicht das Schlüsselwort `case` wie bei den anderen Alternativen und des Weiteren ist auch nicht die Verwendung der `break`-Anweisung notwendig (da das `switch`-Konstrukt danach ohnehin automatisch beendet wird).[45]

Wiederholungen

Grundsätzlich werden in der prozeduralen Programmierung zwei unterschiedliche Konstrukte eingesetzt, um Befehle wiederholt auszuführen – Iterationen und Rekursionen. Da für das Projekt und die Unterrichtsszenarien nur erstere von Bedeutung sind, werden diese nun

[43] vgl. Reimers und Thies 2012, S. 117.
[44] Reimers und Thies 2012, S. 117.
[45] vgl. Reimers und Thies 2012, S. 117-119.

erörtert, letztere jedoch ausgespart, da eine Beschäftigung damit den Rahmen dieser Arbeit sprengen würde.

Bei Iterationen handelt es sich um Schleifen. Bei allen Arten von Iterationen gibt es für die Ausführung der Iteration einen Wert (in der Regel einen sogenannten Laufindex) anhand dessen entschieden werden kann, ob die Iteration ein weiteres Mal durchlaufen oder abgebrochen werden soll. PHP unterstützt mehrere Arten von Iterationen: `while`-Schleifen, `do-while`-Schleifen, `for`-Schleifen und `foreach`-Schleifen.

`while`-Schleifen: Bei dieser Art von Schleife ist die Anzahl der Durchläufe im Vorhinein nicht explizit bekannt, stattdessen wird die Schleife so lange durchlaufen, wie eine gesetzte Bedingung im booleschen Sinn `true` ist. Vor jedem neuen Schleifendurchlauf wird die Gültigkeit der Bedingung erneut überprüft. Sobald sie schließlich zum ersten Mal zu `false` ausgewertet wird, wird die Schleife beendet.[46] Im Folgenden (Abb. 7) ist nun der Aufbau einer `while`-Schleife ersichtlich:

```
while($C)
{
	// Anweisungen X ausführen
}
```
[47]

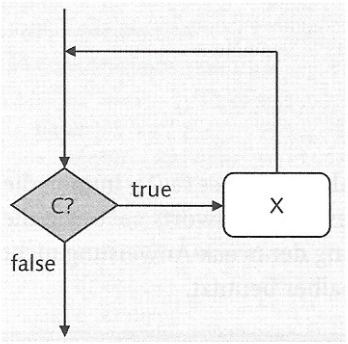


Abbildung 7 Struktur einer while-Schleife

Die Parameter der Bedingung können im Schleifenrumpf abgeändert werden, was wiederum den Abbruch steuert. So summiert der folgende Code beispielsweise alle Zahlen von 0 bis 3:

```
$zaehler = 0;
$ergebnis = 0;
while($zaehler <= 3)
{
	$ergebnis = $ergebnis + $zaehler;
	$zaehler++;
}
```
[48]

Vor dem Schleifenrumpf wird ein Zähler initialisiert, der als Schleifenbedingung geprüft wird. Zusätzlich benötigt die `while`-Schleife eine externe Variable `$ergebnis`, in der die

[46] vgl. Reimers und Thies 2012, S. 119.
[47] Reimers und Thies 2012, S. 120.
[48] Reimers und Thies 2012, S. 120.

addierte Summe zwischengespeichert wird. Vor jedem neuen Schleifendurchlauf wird kontrolliert, ob der Wert der Variablen `$zaehler` kleiner oder gleich 3 ist – nur wenn dies der Fall ist, wird der Wert von `$zaehler` zu `$ergebnis` addiert. Bevor eine neue Iteration beginnt, muss der Zähler angepasst werden, damit keine Endlosschleife entsteht. Sobald der Zähler schließlich durch `$zaehler++;` den Wert 4 erhält, ist die Bedingung nicht mehr erfüllt und die Schleife wird abgebrochen. In der Variable `$ergebnis` ist dann die Summe aller Zahlen von 0 bis 3 gespeichert, in diesem Fall also 6.[49]

Es ist aber auch möglich, einen PHP-Befehl als Bedingung einer `while`-Schleife zu benutzen, welcher boolesche Wahrheitswerte als Antworttypen hat. In Zusammenhang mit MySQL-Datenbanken trifft dies u.a. auf `mysql_fetch_array()` zu, mit dem das Ergebnis einer Datenbankabfrage wie folgt zeilenweise in ein PHP-Array übertragen wird: `while($row=mysql_fetch_array($result)) { ... }`[50]

Mit jedem Aufruf der Schleife wird ein neuer Datensatz des Datenbankergebnisses – in diesem Beispiel `$result` – im Array `$row` gespeichert. Im Schleifenrumpf kann der Inhalt von `$row` dann verarbeitet und beispielsweise ausgegeben werden. Sind alle Daten auf diese Weise einmal durchlaufen, gibt `mysql_fetch_array()` den Wert `false` zurück und die `while`-Schleife wird daraufhin beendet. Gibt die Datenbank aber z.B. eine leere Menge als Ergebnis zurück, so wird die Schleife gar nicht erst ausgeführt, sondern von vornherein abgebrochen. Dies ist jedoch nicht in allen Fällen wünschenswert – eine Lösung bieten folgende Schleifen an.[51]

`do-while`-Schleifen: Möchte man, dass die Schleife mindestens einmal ausgeführt wird, so kann das sogenannte `do-while`-Konstrukt gewählt werden. Dieses prüft die Bedingungen nämlich erst *nach* dem Schleifendurchlauf. Bis auf die Auswertungsreihenfolge des Rumpfes und der Bedingung reagiert eine `do-while`-Schleife aber exakt wie eine `while`-Schleife. Im Folgenden wird nun das Grundgerüst einer `do-while`-Schleife gezeigt: [52]

```
do
{
      // Anweisungen X ausführen
}
while($C)
```
[53]

Beide Schleifen lassen sich durch ein `break;` beenden. Sinnvoll kann dies u.a. dann sein, wenn die Schleife als endlos entworfen ist, wie an folgendem Code und Abb. 8 ersichtlich:

[49] vgl. Reimers und Thies 2012, S. 120.
[50] vgl. Reimers und Thies 2012, S. 120.
[51] vgl. Reimers und Thies 2012, S. 120.
[52] vgl. Reimers und Thies 2012, S. 121.
[53] Reimers und Thies 2012, S. 121.

```
while(true)
{
	// diverse Anweisungen ausführen
	// Schleife unter bestimmten Umständen abbrechen
	break;
}
```

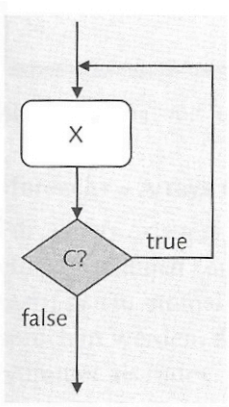


Abbildung 8 Struktur einer do-while-Schleife

`for`-Schleifen mit Laufindex: Die Parameter einer `for`-Schleife werden, anders als bei einer `while`-Schleife, im Kopf definiert. Dazu gehören zum einen die Initialisierung einer Laufvariablen (`A`), als auch eine Abbruchbedingung (`C`) und zum anderen eine Anweisung zur Manipulation der Laufvariablen (`B`), wie an folgendem Code ersichtlich:[54]

```
for (A; C; B)
{
	// Anweisungen X ausführen
}
```

Vor dem ersten Durchgang wird ein Laufindex initialisiert (`A`), welcher auch innerhalb des Schleifenrumpfs zur Verfügung steht. Die Abbruchbedingung (`B`) muss nicht zwingend Bezug auf den Laufindex nehmen, aber für gewöhnlich ist dies der Fall. Solange die Bedingung erfüllt ist, welche vor jedem Durchlauf der Schleife neu überprüft wird, startet der nächste Durchlauf. Nachdem der Rumpf komplett abgearbeitet wurde, wird der Index (normalerweise) verändert (`C`).[55] An einem Beispiel sieht dies wie folgt aus:

```
for ($i=1; $i <= 10; $i++)
{
	echo 'Der Laufindex i hat gerade folgenden Wert: '.$i.'<br>';
}
```
[56]

Als Ergebnis erhält man eine Ausgabe von zehn Zeilen, aus denen jeweils der aktuelle Wert der Laufvariablen abzulesen ist (von 1 bis 10).

Wenn die Parameter der Schleife nicht gut aufeinander abgestimmt sind, kann es zu einer Endlosausführung kommen. Dies kann z.B. passieren, wenn man – auf das vorige Beispiel

[54] vgl. Reimers und Thies 2012, S. 122.
[55] vgl. Reimers und Thies 2012, S. 122.
[56] Reimers und Thies 2012, S. 122.

bezogen – den Laufindex dekrementiert, anstatt ihn zu inkrementieren, denn dann würde die Abbruchbedingung immer erfüllt sein.[57]

Eine `for`-Schleife eignet sich aufgrund der Verwendung von Laufvariablen besonders gut, um Arrays mit fortlaufendem Index abzuarbeiten. So kann beispielsweise im Rumpf dann mit `$array[$i]` gezielt auf ein Element zugegriffen werden. Als Abbruchbedingung muss man definieren, dass die Schleife so lange laufen soll, wie der Laufindex kleiner als die Anzahl der Elemente des Arrays ist. Wichtig ist, dass die Bedingung kleiner (<) verwendet wird und nicht kleiner gleich (≤), da das erste Element eines Arrays den Index 0 hat. Der höchste Index ist somit um 1 kleiner als die Anzahl der Elemente.[58] Es folgt nun ein Beispiel, um dies zu verdeutlichen:

```
$testArray = array(1,5,9,15,19);
for ($i=0; $i < count($testArray); $i++)
{
      echo $testArray[$i].'<br>';
}
```
[59]

`foreach`-Schleifen: Mit `foreach` lassen sich Arrays durchlaufen, ohne dass man einen Laufindex definieren muss. Die Schleife arbeitet das angegebene Array der definierten Reihenfolge nach ab und setzt nach jedem Durchlauf den internen Zeiger des Arrays auf das jeweils folgende Element. Das aktuelle Schlüssel-Wert-Paar des Arrays ist an den Variablen gebunden, welche im Kopf der Schleife definiert werden.[60] Folgender Code befüllt ein assoziatives Array und gibt dies mit Hilfe einer `foreach`-Schleife aus:

```
$assoziativesArray = array();
$assoziativesArray['eins'] = 'erster Wert';
$assoziativesArray['zwei'] = 'zweiter Wert';
$assoziativesArray['drei'] = 'dritter Wert';

foreach($assoziativesArray as $key=>$value)
{
      echo 'Wert des Arrayfeldes ';
      echo '<b>'.$key.'</b>: <i>'.$value.'</i></b>';
}
```
[61]

[57] vgl. Reimers und Thies 2012, S. 123.
[58] vgl. Reimers und Thies 2012, S. 123.
[59] vgl. Reimers und Thies 2012, S. 123.
[60] vgl. Reimers und Thies 2012, S. 124.
[61] Reimers und Thies 2012, S. 123-124.

Dieser Codeabschnitt erzeugt folgende Ausgabe:

Wert des Arrayfeldes **eins**: *erster Wert*
Wert des Arrayfeldes **zwei**: *zweiter Wert*
Wert des Arrayfeldes **drei**: *dritter Wert*

3.1.8 Vordefinierte Informationen

Wie bereits in der Einleitung dieses Kapitels in Bezug auf PHP angesprochen, bietet PHP zahlreiche hilfreiche Konstrukte, mit denen man unkompliziert arbeiten kann. So enthält beispielsweise die lokale PHP-Umgebung zahlreiche Variablen, Konstanten und Funktionen, auf die man beim Programmieren immer wieder zurückgreifen kann.[62]

Superglobale Arrays

Prominente Vertreter der vordefinierten Variablen sind die sogenannten superglobalen Arrays, welche u.a. Systeminformationen des Webservers, Sitzungsdaten der Benutzer oder Werte aus HTML-Formularen enthalten. Der Begriff „superglobal“ bedeutet, dass die Daten in jedem Gültigkeitsbereich nutzbar sind (auch ohne das Schlüsselwort global).[63]

Arrays aus Benutzereingaben: Benutzereingaben können über ein HTML-Formular in das Skript gelangen. So ist PHP nämlich in der Lage, sämtliche Formularkonstrukte (wie z.B. ein- und mehrzeilige Eingabefelder, ein- und mehrwertige Auswahlboxen, Auswahllisten mit Ein- und Mehrfachauswahl) in Variablen zu übernehmen. Je nach verwendeter Übertragungsmethode und Inhalt werden verschiedene Arrays angelegt, welche nun im Folgenden kurz beschrieben werden.[64]

`POST`: Das Array `$_POST` enthält alle Werte, die über die gleichnamige Methode übermittelt wurden – der Code lautet hierzu wie folgt: `<form action="" method="POST"></form>`[65]

`$_POST` ist ein sogenanntes assoziatives Array. Die Namen der Arrayfelder resultieren bei der Verwendung eines Formulars aus den `name`-Attributen der Formularbestandteile. Aus der Anmeldeoberfläche aus folgendem Code

```
<form action="" method=POST">
      <input type="text" name="username">
      <input type="password" name="pw">
      <input type="submit" name="login" value="Einloggen">
</form>
```
[66]

[62] vgl. Reimers und Thies 2012, S. 129.
[63] vgl. Reimers und Thies 2012, S. 130.
[64] vgl. Reimers und Thies 2012, S. 131.
[65] Reimers und Thies 2012, S. 131.
[66] vgl. Reimers und Thies 2012, S. 131.

wird von PHP ein entsprechendes Array erzeugt, dessen Ausgabe mit `print_r ($_POST);` wie folgt aussieht:

```
Array
(
[username] => Name
[pw] => Passwort
[login] => Einloggen
)
```[67]

GET: Das Array `$_GET` unterscheidet sich von `$_POST` nur insofern, dass hier die Daten mit der `GET`-Methode vom Client an den Server verschickt werden. Daten lassen sich somit ebenso einfach mit einem (X)HTML-Formular versenden: `<form action="" method="GET"> </form>`[68]

Daten per Hand an ein Skript zu übergeben, ist mit der `GET`-Methode leichter als mittels `POST`, da die Parameter zusammen mit der URL vom Client an den Server übermittelt werden. Dafür ist `GET` nicht dafür geeignet, Passwörter sicher zu übertragen – weil nämlich `GET`-Parameter in der URL des aktuellen Skriptes enthalten sind und dadurch Passwörter vom Bildschirm oder aus dem Browsercache abgelesen werden können.[69]

SESSION: Für die Realisierung von Sitzungen kann man entweder Cookies verwenden oder auf die Session-Variablen zurückgreifen. Da ich mich in Bezug auf meine Diplomarbeit für letztere Möglichkeit entschieden habe, möchte ich diese hier beschrieben: Um das Skript mit einer Sitzung zu versehen, muss man am Anfang des Codes den Befehl `session_start()` ausführen. Wenn keine Session besteht, wird dadurch eine neue erzeugt – anderenfalls wird die alte Sitzung fortgeführt. Alle in der Sitzung gespeicherten Werte liegen dann im Array `$_SESSION`. Um eine Session zu beenden, z.B. bei einem Logout, wird der Befehl `session_destroy()` genutzt.[70] Ein typisches Logout-Skript sieht wie folgt aus:

```
session_start();
$_SESSION = array();
session_destroy();
echo 'Sie haben sich ausgeloggt.';
```[71]

Bevor die Sitzung beendet wird, werden alle darin gespeicherten Werte gelöscht. Dies geschieht, indem man `$_SESSION` – wie im Code ersichtlich – ein leeres Array zuordnet. Die Ausgabe mit `echo` am Ende dient schließlich nur der Benachrichtigung des Benutzers, hat ansonsten keinen weiteren Zweck.[72]

---

[67] vgl. Reimers und Thies 2012, S. 131-132.
[68] vgl. Reimers und Thies 2012, S. 132.
[69] vgl. Reimers und Thies 2012, S. 132.
[70] vgl. Reimers und Thies 2012, S. 136.
[71] Reimers und Thies 2012, S. 137.
[72] vgl. Reimers und Thies 2012, S. 137.

Ob eine Session existiert oder nicht, erfährt man durch die Funktion `session_start()`, welche eine Zahl zwischen 0 und 2 zurückgibt, die die folgenden Werte repräsentiert: 0: `PHP_SESSION_DISABLED` (Sessions sind prinzipiell deaktiviert), 1: `PHP_SESSION_NONE` (es ist keine Session gestartet), 2: `PHP_SESSION_ACTIVE` (es ist eine Session aktiv).[73]

3.1.9 Einbinden externer Dateien

Wie bereits im Kapitel „3.1.6 Funktionen" angesprochen, ist Kapselung ein gutes Mittel, um Struktur in den Code zu bringen. Auf einer höheren Ebene – nämlich in Bezug auf Skripte und externe Dateien – kann man mit der gleichen Zielsetzung Modularisierung nutzen und Code in logischen Blöcken auf mehrere Dateien aufteilen. So können beispielsweise Kernfunktionen in eine externe Datei ausgegliedert werden oder aber auch alle datenbankrelevanten Funktionen, wie ich dies auch bei der Programmierung gemacht habe. Es gibt mehrere Möglichkeiten, die Modularisierung umzusetzen – das Ziel sollte jedoch immer darin bestehen, den Überblick zu behalten. Die Vorteile der Modularisierung decken sich mit denen der Kapselung (siehe Unterkapitel Kapselung in „3.1.6 Funktionen"). Modularisierung ist jedoch nicht der einzige Fall, bei dem externe Dateien einzubinden sind, denn auch bei der Verwendung offener Bibliotheken muss man die bereitgestellten Funktionalitäten integrieren.[74]

PHP stellt zwei Befehle zum Einbinden externer Dateien bereit: `include` und `require`. Sie werden im Code wie folgt verwendet: `include('externe_datei.php');` bzw. `require('weitere_datei.php');`. Wie man erkennen kann, unterscheiden sich die beiden Befehle bezüglich ihrer Syntax nicht voneinander. Als Parameter gibt man den Speicherort der einzubindenden Datei in Form eines Strings ein – dies kann entweder eine lokale Datei sein oder auch eine URL. Die Verwendung von Dateien, die auf fremden Servern liegen, kann jedoch durch die Sicherheitseinstellungen von PHP unterbunden sein.[75]

Nun zum Unterschied dieser beiden genannten Befehle: Dieser besteht nämlich darin, wie sie auf fehlerhafte Einbindungen reagieren. Ist die Datei nicht lesbar, dann bekommt man bei `include()` lediglich eine Warnung, das Skript wird jedoch weiter ausgeführt. Der Befehl `require()` ist hierbei restriktiver: Funktioniert das Einbinden nicht, so wird das Skript mit einem Fehler abgebrochen.[76]

[73] vgl. Reimers und Thies 2012, S. 138.
[74] vgl. Reimers und Thies 2012, S. 143.
[75] vgl. Reimers und Thies 2012, S. 143-144.
[76] vgl. Reimers und Thies 2012, S. 144.

Beim Verwenden von Einbindungen kann es zu unterschiedlichen Problemen kommen – eine davon sind die sogenannten Mehrfacheinbindungen, welche in folgender Abbildung veranschaulicht werden:

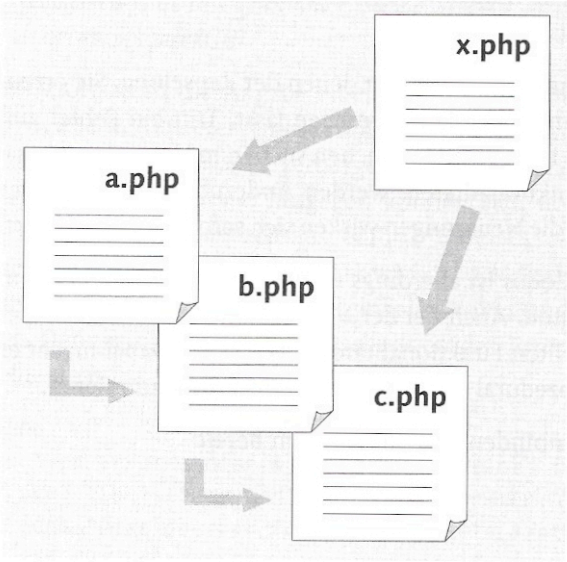


Abbildung 9 Mehrfacheinbindungen führen zu Problemen

In Abb. 9 besteht eine Kette von Einbindungen externer Dateien: Das Skript `a.php` bindet die externe Datei `x.php` ein. `a.php` wird wiederum von `b.php` und dieses von `c.php` eingebunden. Schließlich bindet als letztes `c.php` auch die externe Bibliothek `x.php` ein. Diese Form von Mehrfacheinbindungen führt in den meisten Fällen zu Fehlern, da einerseits die Funktionen und Konstanten von `x.php` doppelt definiert werden, was zu Parse-Fehlern führt, andererseits werden die Variablen aus `x.php`, die über Umwege auch in das Skript `c.php` gelangen durch das erneute Einbinden auf den Wert zurückgesetzt, der in `x.php` gesetzt ist. In diesem Fall wären alle bis dahin eventuell gemachten Verarbeitungen vergebens. Wie geht man am besten mit diesem Problem um? Um Mehrfacheinbindungen zu vermeiden, gibt es in PHP die erweiterten Varianten `require_once()` und `include_once()`. Verwendet man diese, so wird ein Skript, das schon eingebunden war, kein zweites Mal eingebunden, und der Befehl dann einfach übergangen, ohne dass eine Fehlermeldung ausgegeben wird.[77]

[77] vgl. Reimers und Thies 2012, S. 145.

3.2 Datenbank

3.2.1 Allgemeines

Was ist eine Datenbank und wozu wird sie benötigt?

Um diese Frage beantworten zu können, muss ein bisschen ausgeholt werden: Menschen sammeln mit ihren Sinnesorganen Daten, die sie zu Informationen verdichten. Aus diesen Informationen entsteht Wissen, das uns hilft, die Welt zu verstehen und zu kontrollieren. Diese Daten werden in unseren Gehirnen gelagert – unwichtige Daten werden vergessen, wichtige werden aufgeschrieben, damit sie nicht vergessen werden. Damit aus diesen vielen Daten kein unbeherrschbares Datenchaos entsteht, haben sich Menschen immer wieder praktische Hilfsmittel ausgedacht, um Daten zu konservieren, zu strukturieren oder auszuwerten. Dies fing bereits vor der Erfindung der Schrift an und ging von Kartei- und Zettelkästen und Registraturen über Tabelliermaschinen schließlich bis hin zu Computern.[78]

Im Kontext moderner IT wurde im Laufe der Zeit dabei der Begriff „Datenbank“ (vom engl. database) geprägt. Eine Definition davon lautet wie folgt: „Eine Datenbank ist eine Sammlung von Daten, die von einem Datenbankmanagementsystem (DBMS) verwaltet wird.“[79] Die Daten werden innerhalb der Datenbank in sogenannten Datentabellen gespeichert (siehe Abb. 10).

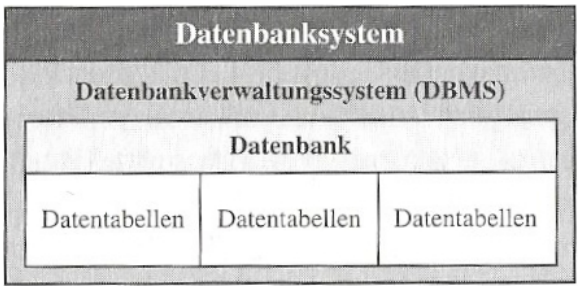


Abbildung 10 DBMS

Datenbankmanagementsystem (DBMS)

Wie bereits angesprochen, benötigt eine Datenbank ein Datenbankmanagementsystem (DBMS). Worum handelt es sich dabei? Das DBMS ist eine Verwaltungssoftware, die folgende Hauptaufgaben hat: Sie soll den Benutzern die Möglichkeit geben, Daten in die Datenbank einzufügen, Daten aus der Datenbank zu löschen, Daten in der Datenbank zu ändern und Daten in der Datenbank zu suchen.[80]

In einer Datenbank werden mehrere Tabellen zusammengefasst. Der Begriff Datenbank ist dabei nicht gleichzusetzen mit Datenbankmanagementsystem (DBMS) – so sind Datenbanken

[78] vgl. Piepmeyer 2011, S. 3.
[79] Piepmeyer 2011, S. 3.
[80] vgl. Piepmeyer 2011, S. 4.

nämlich Bestandteil eines DBMS. Darüber hinaus enthält das DBMS aber weitere wesentliche Merkmale, wie z.B. ein Benutzermanagement für den Mehrbenutzereinsatz, die Verwaltung von Zugriffsrechten und Schnittstellen zu externen Clientsystemen. Der Aufbau eines Datenbanksystems aus der Anwendersicht ist in Abb. 11 ersichtlich.

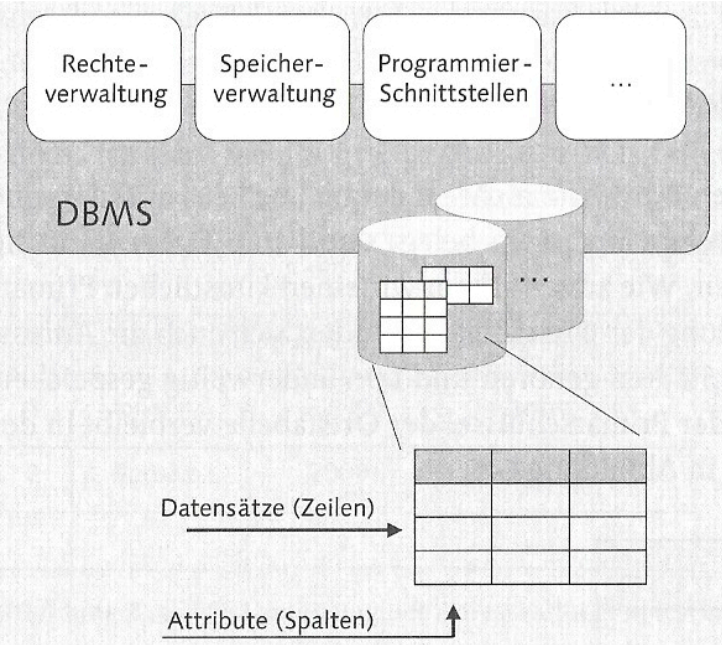


Abbildung 11 Aufbau des Datenbankmanagementsystems

Datenbankadministrator/Datenbankadministratorin

Da ein DBMS in den meisten Fällen noch nicht ganz ohne menschliche Unterstützung auskommt, gibt es den sogenannten Datenbankadministrator bzw. die Datenbank-administratorin, welcher bzw. welche bestimmte Aufgaben hat. So legt dieser bzw. diese beispielsweise die Struktur des Datenbestandes fest, definiert die Integritätsregeln (siehe folgendes Unterkapitel „Konsistenz und Integrität“) der Datenbank und vergibt und entzieht Zugangsberechtigungen. Bei Störungen sollte dieser bzw. diese wissen, was zu tun ist, um das DBMS wieder in den laufenden Betrieb zurückzubringen. Er bzw. sie bildet daher die Schnittstelle zur Supportorganisation des DBMS-Herstellers.[81]

Nun werden Aspekte angesprochen und erörtert, die in Bezug auf eine Datenbank wichtig sind. Dabei handelt es sich um die Folgenden: Konsistenz und Integrität, dauerhafte Speicherung, gleichzeitiger Zugriff von mehreren Benutzern, Zugangsbeschränkungen, Performance, Zuverlässigkeit und Fehlertoleranz.

[81] vgl. Piepmeyer 2011, S. 9.

Konsistenz und Integrität

Grundvoraussetzung für Datenbankmanagementsysteme ist die sogenannte Konsistenz, welche wie folgt definiert wird: „Jede Änderung des Datenbestands überführt die Datenbank von einem logisch korrekten Zustand in einen anderen logisch korrekten.“[82]

Eine Datenbank hilft nur, wenn man sich auf ihren Inhalt verlassen kann. Insbesondere erwartet man sich, dass Daten logisch korrekt sind. Der springende Punkt dabei ist, dass man selbst bestimmt, was „logisch korrekt“ bedeutet. In diesem Zusammenhang ist der Begriff „Konsistenz“ also im semantischen Kontext zu verstehen. Bei Gehältern will man z.B. gewährleisten, dass sie nicht negativ werden. Solche Anforderungen an einen konsistenten Datenbestand formuliert man in Form sogenannter Integritätsregeln – und ein DBMS wiederum überwacht die Einhaltung dieser.[83]

Formuliert man also beispielsweise die Regel, dass es keine negativen Gehälter geben darf, dann kann niemand – auch wenn er noch so viele Rechte hat – mit Hilfe des DBMS eine Operation ausführen, die den Datenbestand so ändert, dass dieser negative Gehälter enthält. Die Menge aller Integritätsregeln definiert die Konsistenz unserer Daten.[84] Ein interessantes Zitat in diesem Zusammenhang lautet wie folgt: „Security means protecting the data against unauthorized users. Integrity means protecting the data against authorized users.“[85]

Sicherheit und Integrität(sregeln) sind also dementsprechend zwei verschiedene Paar Schuhe. Wie diese Regeln formuliert werden, hängt stark vom Datenbanksystem ab. Da sie die Korrektheit der Daten sicherstellen, gelten sie ausnahmslos für alle Anwender, weil ja keine einzige Person die Konsistenz der Daten zerstören darf.[86] Eine wesentliche Aufgabe eines DBMS kann also wie folgt zusammengefasst werden: „Ein DBMS versorgt berechtigte Anwender mit konsistenten Daten.“[87]

Hinsichtlich der Konsistenz des Datenbestandes darf es keine Kompromisse geben, denn ein – wenn auch nur teilweise – inkonsistenter Datenbestand ist wertlos. Wichtig ist in diesem Zusammenhang auch, dass Inkonsistenzen nur durch Änderungen des Datenbestandes entstehen können. Das DBMS muss also immer dann die Konsistenz sicherstellen, wenn Benutzer Daten ändern. Wenn mehrere Anwender mit den Daten arbeiten, müssen auch etwaige Probleme, die sich aus dem gleichzeitigen Zugriff ergeben, berücksichtigt werden.[88]

---

[82] Piepmeyer 2011, S. 4.
[83] vgl. Piepmeyer 2011, S. 7-8.
[84] vgl. Piepmeyer 2011, S. 8.
[85] Piepmeyer 2011, S. 8.
[86] vgl. Piepmeyer 2011, S. 8.
[87] Piepmeyer 2011, S. 8.
[88] vgl. Piepmeyer 2011, S. 8.

Dauerhafte Speicherung

Ein weiterer wichtiger Aspekt von Datenbanken ist die dauerhafte Speicherung. Man unterscheidet zwischen volatilen (flüchtigen) Daten und persistenten (dauerhaften) Daten. Erstere existieren nur während der Laufzeit des Programms, letztere hingegen werden auf Speichermedien, wie z.B. Festplatten, gehalten. So sind persistente nämlich auch dann verfügbar, wenn beispielsweise die Software, mit der sie angelegt wurden, nicht mehr existiert oder der Rechner, auf dem sie erzeugt wurden, ausgeschaltet ist.[89]

In diesem Zusammenhang soll aber angemerkt werden, dass es auch DBMS gibt, bei denen die volatile Datenhaltung praktiziert wird. Der große Vorteil der Datenhaltung im Hauptspeicher ist nämlich, dass sie wesentlich effizienter ist als die Datenhaltung auf Festplatten. Die Ein- und Ausgabe (also der Datentransport zwischen Hauptspeicher und Festplatte) ist ein signifikanter Engpass für persistierende DBMS. Wenn Daten also nicht für die Ewigkeit geschaffen werden, spricht so gesehen auch nichts gegen eine flüchtige Datenhaltung.[90]

Gleichzeitiger Zugriff von mehreren Benutzern

Ebenfalls zu berücksichtigen in Bezug auf Datenbanken bzw. Datenbankmanagementsysteme ist, dass diese mehreren Anwendern gleichzeitig den Zugriff auf Daten ermöglichen können (wie z.B. bei einem Webshop). Aber auch wenn die Mehrbenutzerfähigkeit eine typische Eigenschaft eines DBMS ist, so gibt es doch Datenbanksysteme, die nicht für den Zugriff durch mehrere Benutzer entwickelt wurden.[91]

Einen Überblick über die Struktur eines Client-Server-Systems zeigt Abb. 12:

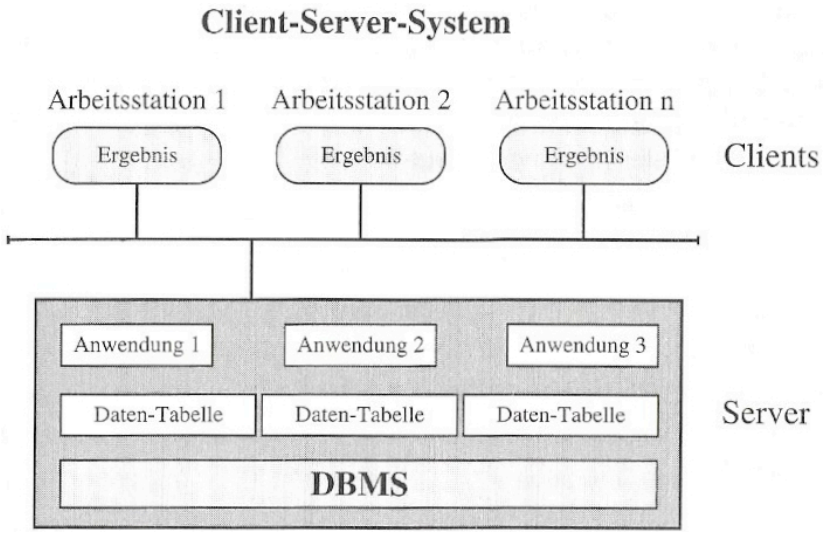


Abbildung 12 Client-Server-System

[89] vgl. Piepmeyer 2011, S. 5.
[90] vgl. Piepmeyer 2011, S. 6.
[91] vgl. Piepmeyer 2011, S. 6.

Zugangsbeschränkungen

In Bezug auf die Sicherheit der verwalteten Daten ist zu sagen, dass ein DBMS diese sicherstellen kann, indem es die Definition feingranularer Zugangsbeschränkungen zu den Daten ermöglicht. Das bedeutet, dass bestimmten Benutzern bestimmte Rechte zugeteilt werden. Diese Granularität der Zugangsbeschränkungen reicht vom vollständigen Ausschluss nicht autorisierter Anwender über spezifische Rechte zum Lesen, Ändern, Einfügen und Löschen von Daten bis hin zur sogenannten „Generalvollmacht", bei welcher der Benutzer das Recht zur unbeschränkten Bearbeitung der Daten der Datenbank erhält.[92]

Performance

Abfragen werden in Bruchteilen von Sekunden abgearbeitet, auch wenn auf einer Datenbank mit Tausenden von Daten gleichzeitig mehrere Benutzer arbeiten. Weitere Verbesserungen in puncto Performance können beispielsweise durch Katalogisierung und Indizierung von Inhalten erreicht werden.[93]

Zuverlässigkeit und Fehlertoleranz

Ein weiterer wichtiger Aspekt von Datenbankmanagementsystemen ist, dass diese zuverlässig und fehlertolerant arbeiten (müssen). So darf es beispielsweise bei Stromausfällen nicht passieren, dass Datensätze nur teilweise auf die Festplatte geschrieben wurden. Die Daten müssen daher nicht nur logisch, sondern auch physikalisch konsistent sein. Idealerweise reagiert das DBMS auf praktisch jeden denkbaren Fehlerfall so, dass der laufende Betrieb stets gewährleistet ist. Für Unternehmen mit beispielsweise einem Webshop ist dies auch äußerst wichtig, da es z.B. zu sofortigen Umsatzeinbußen kommen kann, wenn Kunden nicht auf Artikel zugreifen können. Wenn man bereit ist, genug zu investieren, sodass das DBMS unterbrechungsfrei läuft, kann man ein praktisch beliebig hohes Maß an sogenannter Fehlertoleranz erzielen.[94]

Diese gerade behandelten Aspekte heben ein DBMS recht deutlich von der Konkurrenz ab, wie z.B. der Datenhaltung in Dateien. Das Ablegen in einer Datei ist sehr wohl der intuitivste Weg, Daten dauerhaft zu speichern – Datenbanksysteme machen letztlich auch nichts anderes. Neben der transparenten Speicherung bietet aber ein DBMS weiters auch komfortable Zugriffsmöglichkeiten auf den Inhalt der Datenbanken. Solche Strukturen müsste man bei der Verwendung von Textdateien erst implementieren.[95]

[92] vgl. Piepmeyer 2011, S. 7.
[93] vgl. Reimers und Thies 2012, S. 193.
[94] vgl. Piepmeyer 2011, S. 9.
[95] vgl. Reimers und Thies 2012, S. 194.

In Bezug auf das Projekt wurde ebenfalls mit einer Textdatei gearbeitet und zwar in der Hinsicht, dass die Namenliste (genauere Infos dazu sind im Kapitel „4. Entschlüsselung der Zahlenrätsel von Ephesos“ zu finden) als sogenannte „CSV-Datei“ erhalten wurde. Die Abkürzung „CSV“ steht für für „Comma-separated Values“ und bedeutet, dass einzelne Felder eines Datensatzes mittels Kommas getrennt werden und einzelne Datensätze untereinander mittels Enter (neue Zeile bedeutet neuer Datensatz). Mit Hilfe dieser Datei konnten dann mittels phpMyAdmin diese Werte in die Datenbank importiert und damit gearbeitet werden.

3.2.2 Relationale Datenbanksysteme

MySQL als relationales Datenbanksystem

Das Konzept von relationalen Datenbanksystemen beruht auf dem mathematischen Begriff der Relation. Zu den relationalen Datenbanksystemen gehört auch MySQL, welches noch genauer im folgenden Unterkapitel („3.2.3 MySQL und SQL“) behandelt wird. Nun zu einigen Begriffserklärungen: Eine Relation ist eine Beziehung zwischen zwei oder mehreren Objekten – im Kontext einer Datenbank stellen Relationen zweidimensionale Tabellen dar. Jede Tabellenzeile, ein sogenanntes Tupel, repräsentiert eine sogenannte Entität, ein eindeutiges Objekt. Beispiele für Entitäten wären z.B. ein einzelner Kontakt aus einer Adresskartei, ein bestimmter Eintrag in einem Gästebuch oder ein auf einer Auktions-Plattform angebotener Auktionsartikel mit Artikelnummer.[96]

Gleiche Entitäten werden zu einem Entitätstyp zusammengefasst. Bezogen auf die ebengenannten Beispiele wären dies „Kontakte“, „Einträge“ und „Artikel“. Datenbank-tabellen tragen meist den Namen eines Entitätstyps. Die Tabellenspalten wiederum beschreiben die Attribute, die alle Entitäten gemeinsam haben (hier in Kleinbuchstaben). So kann beispielsweise ein Kontakt aus den Attributen „Identifikationsnummer“ (ID), „Name“, „Postleitzahl“ und „Wohnort“ bestehen.[97] Daraus ergibt sich dann folgende Tabelle (Tabelle 1 – natürlich beliebig erweiterbar bezüglich Spalten und Zeilen):

id	name	postleitzahl	ort
1	Anna	1060	Wien
2	Lisa	2020	Hollabrunn
...	...	...	...

Tabelle 1 Datenbanktabelle, um einen Kontakt zu speichern (Name, Postleitzahl und Ort)

[96] vgl. Reimers und Thies 2012, S. 198.
[97] vgl. Reimers und Thies 2012, S. 198.

Attribute oder Kombinationen von Attributen, die jede Entität eindeutig identifizieren, werden Primärschlüssel genannt. Existiert für die Entitäten eines Typs kein solcher, so wird er künstlich geschaffen. Dieses Attribut vergibt eine fortlaufende Nummer für jeden Datensatz. Sinnvoll ist das allerdings nur, wenn es kein „natürliches“ Gegenstück gibt. Bezogen auf die Adresskartei könnte man denken, dass der Name ein Primärschlüssel ist – da es jedoch mehrere Annas, Lisas usw. gibt, reicht dieses Attribut alleine nicht aus. Der Primärschlüssel müsste also entweder aus mehreren Attributen zusammengesetzt sein oder künstlich hinzugefügt werden. In diesem Beispiel wurde mittels ID die zweite Variante gewählt. Den Primärschlüssel einer Relation erkennt man in dieser Tabelle sofort daran, dass sein Attributname unterstrichen ist.[98] Manchmal wird dieser auch mit einem Schlüsselsymbol dargestellt.

Große Bedeutung bekommt das Konzept von Primärschlüsseln, wenn Tabellen logisch miteinander verknüpft werden. Auf das Beispiel mit der Adresskartei bezogen, würden sich hier z.B. alle Daten, die sich auf den Ort beziehen, in eine zweite Tabelle auslagern lassen. Das betrifft neben Stadtnamen auch die Postleitzahlen.[99] Eine solche Tabelle könnte wie folgt aussehen (Tabelle 2):

id	postleitzahl	name
1	1060	Wien
2	2020	Hollabrunn
...	...	...

Tabelle 2 Datenbanktabelle, um einzelne Orte zu speichern

Warum in dieser Tabelle die Postleitzahl nicht zum Primärschlüssel gemacht wurde, liegt daran, dass diese den Datensatz nicht eindeutig identifiziert, da ein und derselbe Wert in unterschiedlichen Ländern mehrfach belegt sein kann. Daher wird erneut die ID als künstlicher Primärschlüssel eingeführt. Durch die Erstellung der Ortstabelle verändert sich auch die Adresskartei – denn so werden alle Daten, die zu den Städten gehören und nun anderweitig gespeichert sind, entfernt.[100] Nur der Primärschlüssel der Ortstabelle verbleibt in der Adresskartei, wie an folgender Tabelle (Tabelle 3) zu sehen:

id	name	ortRef
1	Anna	1
2	Lisa	2
...	...	...

Tabelle 3 Datenbanktabelle, um eine Person mit zugehörigem Ortseintrag zu speichern (Fremdschlüssel)

[98] vgl. Reimers und Thies 2012, S. 198-199.
[99] vgl. Reimers und Thies 2012, S. 199.
[100] vgl. Reimers und Thies 2012, S. 199.

Über den Primärschlüssel der Ortstabelle lassen sich sämtliche Wohnortangaben referenzieren. Der referenzierte Primärschlüssel wird zum sogenannten Fremdschlüssel. Einen der Vorteile, den die Aufteilung in mehrere Tabellen bietet, ist u.a. die einfache Wartbarkeit. Mit Fremdschlüsseln einher geht die referentielle Integrität. Dies bedeutet, dass sobald Tabellen über Schlüsselbeziehungen miteinander verbunden werden, sichergestellt sein muss, dass alle referenzierten Datensätze erhalten bleiben, solange mindestens eine Referenz besteht. Auf das Adresskartei-Beispiel bezogen würd dies heißen, dass der Datensatz „`2;2020;Hollabrunn`“ nicht gelöscht werden darf, wenn in der Kontakttabelle noch „`Lisa`“ gespeichert ist. Geht die Zuordnung verloren, beeinträchtigen fehlende Daten die Datenbankebene (als auch die Applikationsebene). MySQL garantiert referentielle Integrität nur in einigen Fällen (dies ist abhängig von der Wahl der Storage Engine).[101]

3.2.3 MySQL und SQL

Allgemeines zu MySQL

MySQL ist ein relationales Datenbankmanagementsystem (wie bereits angesprochen). Es gilt sogar als das populärste, quelloffene (Open Source) SQL-Datenbankmanagementsystem der Welt. Open Source bedeutet, dass jeder, der möchte, die Software kostenlos benutzen und verändern darf. Man kann sich auch den Quellcode ansehen und diesen nach eigenen Wünschen abwandeln. MySQL verwendet die Lizenz GPL (GNU General Public License), welche festlegt, was man in unterschiedlichen Fällen mit der Software machen darf und was nicht. Der MySQL Server arbeitet als Client-Server-Version (siehe Abb. 12) und in eingebetteten Systemen.[102]

SQL als Datenabfragesprache

Zuallererst ist es wichtig zu wissen, wofür das Kürzel SQL überhaupt steht – so bedeutet es nämlich *Structured Query Language* und es handelt sich dabei um eine sehr weit verbreitete Datenbanksprache im Bereich der relationalen Datenbanken. Sie gehört zur Familie der deklarativen Sprachen, d.h. der Anwender beschreibt die zu lösende Aufgabe in einem SQL-Kommando und bringt dieses zur Ausführung. Die unterschiedlichen Befehle werden von MySQL in drei Gruppen eingeteilt: Daten-Definitions-Kommandos, Daten-Manipulations-Kommandos und Datenbank-Administrations-Kommandos.[103]

Die *Daten-Definitions-Sprache* ist dazu da, um Strukturen zu bearbeiten. Ausgehend von einem Datenmodell können also Datenbanken und Tabellen angelegt, verändert oder gelöscht

[101] vgl. Reimers und Thies 2012, S. 199-200.
[102] vgl. MySQL AB 2007, S. 18-20.
[103] vgl. Reimers und Thies 2012, S. 201.

werden. Zu den Strukturen von Datenbanken gehören Zeichensatz sowie die Sortierfolge (engl. „collation“), welche die Sortierreihenfolge der gespeicherten Daten festlegt. Für Tabellen sind u.a. die Attribute, deren Datentypen und Größe sowie der Primärschlüssel anzugeben.[104]

Mit der *Daten-Manipulations-Sprache* greift man auf den Datenbankinhalt zu. Diese umfasst neben Einfügen, Ändern und Löschen auch das Abfragen der Daten. „Manipulation“ von Daten ist nicht im negativen Sinn gemeint – es bedeutet in diesem Zusammenhang vielmehr Veränderung.[105]

Die *Daten-Administrations-Sprache* bearbeitet Parameter, die von genereller Bedeutung für das Datenbankmanagementsystem sind, wie z.B. die Zugriffsrechte der Benutzer oder Systemvariablen.[106]

Grundlegende SQL-Kommandos

Im Folgenden werden grundlegende SQL-Kommandos jeweils kurz beschrieben. Zuvor noch ein Hinweis bezüglich des Schreibstils: Alle SQL-Befehle werden in dieser Ausarbeitung gänzlich großgeschrieben und alle weiteren Bestandteile durchgängig klein. Optionale Parameter werden in eckige Klammern gesetzt und Alternativen werden mit einer Pipe (`|`) voneinander getrennt. Zeilenumbrüche dienen nur der Übersichtlichkeit, sind also nicht Bestandteil der SQL-Syntax und ändern nichts an der Funktionsweise.

Es werden nun sieben wichtige SQL-Befehle angesprochen und erklärt. Es handelt sich dabei um: `CREATE`, `INSERT`, `SELECT`, `UPDATE`, `DELETE`, `ALTER` und `DROP`.

`CREATE`: Mit diesem Befehl lassen sich sowohl Datenbanken als auch Tabellen erzeugen. Damit gehört `CREATE` zur Daten-Definitions-Sprache. Eine Datenbank kann wie folgt erstellt werden:

```
CREATE DATABASE [IF NOT EXIST] dbName
[DEFAULT] CHARACTER SET zeichensatz
[DEFAULT] COLLATE sortierung
```
[107]

Innerhalb einer Serverinstanz trägt jede Datenbank einen eindeutigen Namen, welcher nicht länger als 64 Zeichen sein darf und u.a. keinen Slash („/“) und/oder Backslash („\“) beinhalten darf (da diese Bestandteile von Pfadangaben sind). Die optionale Phrase `IF NOT EXISTS` dient dazu, einen Fehler abzufangen, wie z.B. wenn es bereits eine Datenbank mit dem angegebenen Namen gibt (dann wird diese nämlich nicht erneut angelegt). Ein Zeichensatz gibt an, welche Zeichen (also z.B. Zahlen, Buchstaben, Sonderzeichen) für den

[104] vgl. Reimers und Thies 2012, S. 201.
[105] vgl. Reimers und Thies 2012, S. 201.
[106] vgl. Reimers und Thies 2012, S. 202.
[107] Reimers und Thies 2012, S. 210.

Inhalt der Datenbank verwendet werden dürfen. Standard ist hier der Zeichensatz `latin1`. Unter Collation ist eine Reihe von Regeln zu verstehen, welche die Sortierreihenfolge in einer Menge von Daten vorgibt (standardmäßig wählt MySQL das schwedische `latin1_swedish_ci`).[108]

Beim Anlegen einer Tabelle muss jede Spalte genau definiert werden. Allerdings ist es genauso möglich, eine leere Tabelle zu erzeugen. Im Nachhinein lassen sich nämlich jederzeit beliebig viele Attribute hinzufügen. Die grundlegende Syntax dafür lautet wie folgt:

```
CREATE [TEMPORARY][dbName.] TABLE [IF NOT EXIST] tblName
[(attrDef, attrDef ...)]
[PRIMARY KEY (attrName, ...)]
[[DEFAULT] CHARACTER SET zeichensatz]
[COLLATE sortierung]][109]
```

Mit `[TEMPORARY]` gekennzeichnete Tabellen werden automatisch gelöscht, sobald die Datenbankverbindung beendet wird – sie sind also nur sichtbar für die eigene Verbindung. Das bedeutet, dass sie im Mehrbenutzerbetrieb nicht für andere Anwender zugänglich gemacht werden können und somit auch nicht mit gleichnamigen temporären Tabellen anderer User kollidieren. Für Tabellennamen bestehen die gleichen Richtlinien und Begrenzungen wie für Datenbanken. Die Option `[PRIMARY KEY()]` wird benutzt, wenn ein zusammengesetzter Primärschlüssel gebildet werden soll. Bei Schlüsseln, die hingegen nur aus einem Attribut bestehen, kann dies in der entsprechenden Definition angegeben werden. Tabellen können eigene Vorgaben zum verwendeten Zeichensatz und Sortierung enthalten (dadurch werden die globalen Einstellungen der Datenbank überschrieben).[110]

Die Definition eines Attributs besteht aus:

```
attrName datentyp [NOT NULL | NULL]
[DEFAULT standardwert]
[AUTO_INCREMENT]
[UNIQUE [KEY] | [PRIMARY] KEY][111]
```

Mit der Angabe `[NOT NULL]` wird gesteuert, ob eine Entität für dieses Attribut nullwertig sein darf. Da MySQL auch Zahlen als Datentypen unterstützt, ergibt sich die gleiche Problematik wie bei PHP: Die numerische 0 ist ein Wert genau wie 1 oder 100 und somit *nicht* gleichbedeutend mit `NULL`. Dementsprechend kann 0 nicht dafür verwendet werden, um anzuzeigen, dass ein Attribut keinen Wert besitzt. Möchte man sicherstellen, dass bei jeder Einfügeoperation ein Wert angegeben werden muss, so sollte man die Attribute mit der Option `NOT NULL` definieren. MySQL führt dann eine Überprüfung auf fehlende Angaben

[108] vgl. Reimers und Thies 2012, S. 210-211.
[109] Reimers und Thies 2012, S. 211.
[110] vgl. Reimers und Thies 2012, S. 211-212.
[111] Reimers und Thies 2012, S. 212.

durch und gibt eine Meldung aus. Mittels `[DEFAULT wert]` kann ein Standardwert vergeben werden, auf den immer dann zurückgegriffen wird, wenn bei einer Einfügeoperation keine Angabe zu diesem Attribut gemacht wird. Auf diese Weise lässt sich auch sicherstellen, dass die Bedingung `NOT NULL` eingehalten wird. Der Standardwert muss hierbei natürlich zu dem vergebenen Datentyp passen. Die Option `AUTO_INCREMENT` wird wiederum gesetzt, um eine fortlaufende Nummerierung des Attributs sicherzustellen. Bei einer Einfügeoperation ist der Wert einer `AUTO_INCREMENT`-Spalte immer um genau 1 größer als das bislang benutzte Maximum. Diese Option eignet sich also nur für ganzzahlige numerische Typen. Eingesetzt wird sie u.a. bei Primärschlüsseln. Jede Tabelle darf maximal eine `AUTO_INCREMENT`-Spalte besitzen, für die kein Standardwert vergeben werden darf. Kennzeichnet man ein Attribut als `UNIQUE`, so werden doppelte Ausprägungen ausgeschlossen (eine Ausnahme bilden dabei Nullwerte). Die Bedingung der Eindeutigkeit ist implizit erfüllt, wenn eine Spalte durch den Zusatz `[PRIMARY KEY]` als Primärschlüssel definiert wird (für den Primärschlüssel sind Nullwerte ausgeschlossen).[112]

`INSERT`: Einfügeoperationen sind nur für Datensätze in Tabellen definiert. Der Befehl `INSERT` wird somit den Daten-Manipulations-Befehlen zugerechnet. Eine Variante zum Einfügen von Daten lautet wie folgt:

```
INSERT [IGNORE] INTO tblName
[(attrName [, attrName ...])]
VALUES (ausdr | DEFAULT [, ausdr | DEFAULT ...])
[, (ausdr | DEFAULT [, ausdr | DEFAULT ...])]
```
[113]

Die Angabe `IGNORE` sorgt dafür, dass Fehlermeldungen unterdrückt werden. Als `tblName` muss der Name einer bereits existierenden Tabelle angegeben werden. Des Weiteren stehen die Attribute, die belegt werden sollen, getrennt von ihren neuen Werten in einer geordneten Liste unter dem Tabellenamen. Dabei wird die Liste in runde Klammern eingefasst. Hinter `VALUES` folgt eine zweite Liste, welche die Werte enthält. Wichtig ist, dass die Ordnung der beiden Listen identisch sein muss; jedoch muss sie nicht zwingend in derselben Reihenfolge sein, die bei der Tabellendefinition verwendet wurde. Es können auch mehrere Datensätze gleichzeitig eingefügt werden. Nicht jedes Feld muss befüllt werden – so können z.B. Spalten ausgelassen werden, die mit der Option `NULL`, `DEFAULT` oder `AUTO_INCREMENT` definiert wurden. Alle Werte, egal welchen Datentyps, können zum Einfügen in Anführungsstriche gesetzt werden; bei Zeichenketten ist dies obligatorisch, aber auch Zahlen verlieren dadurch nicht ihren Typ.[114]

[112] vgl. Reimers und Thies 2012, S. 212-213.
[113] Reimers und Thies 2012, S. 214.
[114] vgl. Reimers und Thies 2012, S. 214-215.

SELECT: Mittels SELECT realisiert MySQL die Datenabfrage. Die Syntax dieses Befehls ist sehr komplex, da vieles damit gemacht werden kann. So kann man beispielsweise mehrere Tabellen miteinander verbinden, Daten gruppieren oder aggregieren und das Ergebnis in eine Datei schreiben. Der Rückgabewert kann ein einzelner Wert, eine Zeile von Werten oder eine Menge von Zeilen sein. Wichtig ist, dass das Ergebnis einer SELECT-Abfrage unabhängig von der Datenmenge in Tabellenform zurückgegeben wird. Des Weiteren eignet sich SELECT auch dafür Berechnungen durchzuführen, ohne dass sich die Abfrage beispielsweise auf eine Tabelle der Datenbank beziehen muss, wie z.B. bei: SELECT 1+100;.[115]

Die Syntax des SELECT-Befehls sieht wie folgt aus:

```
SELECT [DISTINCT] attrName [, attrName] |
[INTO OUTFILE 'dateiname']
[FROM tblName [, tblName]
[WHERE bedingung]
[GROUP BY attrName [ASC | DESC] [, ...]]
[ORDER BY attrName [ASC | DESC] [, ...]]
[LIMIT [position, ] zeilenanzahl] ][116]
```

Eine Abfrage an die Datenbank wird in folgenden Schritten ausgewertet: Zuerst bildet MySQL aus allen Tabellen, die durch Komma getrennt in der FROM-Klausel aufgelistet sind, das sogenannte kartesische Produkt. Die Definition des kartesisches Produkts lautet wie folgt:

> Für das kartesische Produkt $A_1 \times A_2 \times \ldots \times A_n$ der Mengen $A_1, \ldots, A_n$ gilt: $A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) \mid a_i \in A_i;\ 1 \leq i \leq n\}$. Das kartesische Produkt besteht also aus allen n-Tupeln, die man in dieser Reihenfolge aus den Elementen der Mengen $A_1, \ldots, A_n$ bilden kann. Die Anzahl der Tupel wächst mit der Faktorenzahl des Produktes und beträgt $|A_1| * |A_2| * \ldots * |A_n|$.[117]

Das bedeutet, dass jeder Datensatz aus Tabelle A mit jedem Datensatz aus Tabelle B kombiniert wird (siehe Abb. 13). (Es existieren mehrere Arten der Verknüpfung, welche alle zum Oberbegriff JOIN gehören, welche in dieser Ausarbeitung leider aufgrund der Komplexität und des Umfangs nicht berücksichtigt werden kann.) Als Nächstes wird die so entstehende Relation anhand der WHERE-Bedingungen ausgewertet. Dadurch wird die Ergebnismenge für gewöhnlich signifikant verkleinert. WHERE-Ausdrücke können aus mehreren Teilen bestehen, welche alle im booleschen Sinn für jeden Datensatz der Ergebnismenge true sein müssen (gängige Bedingungen sind Gleichheit oder Größenvergleiche). Anschließend werden diejenigen Attribute extrahiert, die nach dem SELECT angegeben sind. Ein Stern (*) selektiert beispielsweise alle in der Ergebnismenge vorhandenen Attribute. Am Ende wird das Ergebnis sortiert, geordnet und limitiert, falls

[115] vgl. Reimers und Thies 2012, S. 216.
[116] Reimers und Thies 2012, S. 216.
[117] vgl. Piepmeyer 2011, S. 36.

entsprechende Angaben gemacht wurden, und schließlich wird das Ergebnis zurückgeliefert.[118]

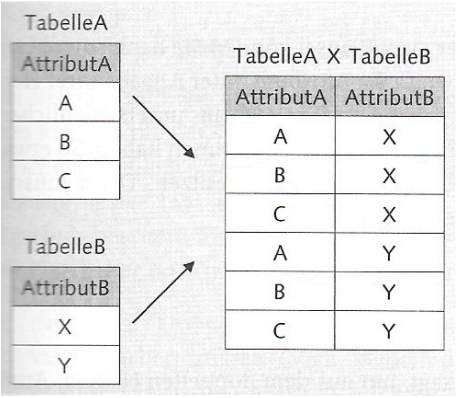


Abbildung 13 Das kartesische Produkt

Nun zu weiteren Parameter, die bei `SELECT` verwendet werden können: `DISTINCT` als optionale Angabe sorgt dafür, dass Zeilen der Ergebnismenge entfernt werden, sobald sie mehrfach auftreten. `GROUP BY` gruppiert die Ergebnismenge nach den angegebenen Attributen, wie z.B. alphabetisch aufsteigend (`ASC` vom engl. „ascending“) oder alphabetisch absteigend (`DESC` vom engl. „descending“). `ORDER BY` sorgt für eine Sortierung der Ergebnismenge (die Reihenfolge ist ebenfalls auf- und absteigend möglich) und der Befehl `LIMIT` schließlich beschränkt die Ergebnismenge in der Anzahl an Datensätzen wie folgt: Mit nur einem Parameter wird angegeben, wie viele Tupel erhalten werden. Ruft man `LIMIT` mit zwei Parametern auf, so gibt der erste an, ab dem wievielten Datensatz die Ausgabe starten soll – der zweite wiederum gibt die Anzahl an.[119]

Ein wichtiger Unterschied zu PHP: Das einfache Gleichheitszeichen („=“) bedeutet in MySQL Zeichengleichheit (bei PHP steht es ja für eine Zuweisung). So würde die `WHERE`-Klausel `name='joh'` kein Ergebnis bringen, wenn man damit z.B. Johann oder Johanna gemeint hätte. Um etwas „unschärfer“ zu suchen, gibt es allerdings folgende Möglichkeit: das Prozentzeichen als Platzhalter („`%`“) im Zusammenhang mit `LIKE`.[120] Es funktioniert wie folgt:

- `name LIKE '%hanna'` bedeutet: Vor der Zeichenkette `hanna` dürfen beliebig viele Zeichen stehen (auch keine) – danach sind keine weiteren Zeichen erlaubt.
- `name LIKE 'Joh%'` bedeutet: Nach der Zeichenkette kann es noch weitere Zeichen geben.
- `name LIKE '%Johann%'` bedeutet: Die Zeichenkette muss irgendwo in `name` vorkommen.[121]

[118] vgl. Reimers und Thies 2012, S. 217.
[119] vgl. Reimers und Thies 2012, S. 217-218.
[120] vgl. Reimers und Thies 2012, S. 219.
[121] vgl. Reimers und Thies 2012, S. 219.

Auch der Unterstrich („_“) kann verwendet werden – dieser ersetzt genau ein Zeichen, wohingegen ja das Prozentzeichen, wie soeben angesprochen, beliebig viele Zeichen ersetzt. Folgende Abbildung (Abb. 14) zeigt eine Liste von Optionen, die in den `WHERE`-Klauseln eingesetzt werden können:

Name	Beispiel (WHERE ...)
= (gleich)	`attribut=1` ist `true` für alle Datensätze, in denen das Attribut den Wert `1` hat.
<=> (null-sicher gleich)	Verhält sich wie der Gleichheitsoperator; einzige Ausnahme: Ergibt `true` (statt `NULL`) beim Vergleich `NULL <=> NULL`.
>= (größer oder gleich)	`attribut>15` ist `true` für alle Datensätze, in denen das Attribut mindestens den Wert `15` hat.
> (echt größer)	`attribut>15` ist `true` für alle Datensätze, in denen das Attribut mindestens den Wert `16` hat.
<= (kleiner gleich)	`attribut<=15` ist `true` für alle Datensätze, in denen das Attribut höchstens den Wert `15` hat.
< (echt kleiner)	`attribut<15` ist `true` für alle Datensätze, in denen das Attribut höchstens den Wert `14` hat.
<> (nicht gleich)	`attribut<>99` ist `true` für alle Datensätze, in denen das Attribut nicht `99` ist.
!= (ungleich)	identisch mit <>
`NOT` (nicht)	Negiert die Aussage der folgenden Vergleichsoperatoren.
`BETWEEN` (zwischen)	`attribut BETWEEN 5 AND 99` ist `true` für alle Datensätze, die einen minimalen Wert von `5` und einen maximalen Wert von `99` haben.
`IS`	`attribut IS TRUE` ist `true` für alle Datensätze, in denen das Attribut im booleschen Sinne `true` ist; möglich ist die Abfrage `IS [NOT] NULL`.
`LIKE`	`attribut LIKE '%wort%'` findet alle Datensätze, in denen `wort` im Attribut vorkommt.
`IN`	`attribut IN (wert1, wert2 ...)` ist `true` für alle Datensätze, bei denen das Attribut einer der Werte `wert1, wert2 ...` ist.

Abbildung 14 Vergleichsoperatoren

Abschließend sei noch angemerkt, dass eine `SELECT`-Anweisung Bestandteil eines anderen SQL-Befehls sein kann. So kann sie beispielsweise in Einfüge-, Aktualisierungs- und Löschoperationen eingefügt werden und kann sogar Teil einer übergeordneten Abfrage sein. Solche Unterabfragen können einen einzelnen Wert, eine Zeile oder sogar eine Menge von Zeilen an die äußere Anweisung übergeben.[122] Folgende Art, Daten in eine Tabelle einzufügen, nutzt die Vorteile einer Unterabfrage:

```
INSERT [IGNORE] INTO tblName
SELECT attrName ...
FROM tblName2 ...
WHERE ...
```
[123]

[122] vgl. Reimers und Thies 2012, S. 221.
[123] Reimers und Thies 2012, S. 221.

UPDATE: Die Syntax des UPDATE-Befehls lautet wie folgt:

```
UPDATE [IGNORE] tblName
SET attrName=ausdr [, attrName=ausdr]
[WHERE bedingung]
[ORDER BY attrName LIMIT n]
```
[124]

Nun zu den möglichen Optionen beim UPDATE-Befehl: Mit der Option IGNORE wird verhindert, dass die Ausführung der Anweisung bei einem Fehler abgebrochen wird. Das ist besonders dann wichtig, wenn mehrere Datensätze auf einmal aktualisiert werden sollen und ein einziger Fehler keinen Effekt auf die Ausführung aller haben soll. Die Spalten, die neu belegt werden sollen, werden einzeln mit direkter Zuweisung nach SET angegeben. Durch die Angabe einer WHERE-Klausel beschränkt man die Aktualisierung auf eine Untermenge des Tabelleninhalts. Die optionale Angabe von ORDER BY hat Einfluss auf die Reihenfolge, in der die Datensätze überschrieben werden. LIMIT begrenzt schließlich die Anzahl der aktualisierten Tupel.[125]

DELETE: Löschoperationen werden mit dem Befehl DELETE vorgenommen. Die Syntax sieht wie folgt aus:

```
DELETE [IGNORE] FROM tblName
[WHERE bedingung]
[ORDER BY attrName LIMIT n]
```
[126]

Wie bereits bei UPDATE angesprochen, schränkt auch hier die WHERE-Klausel die zu löschenden Datensätze in tblName ein. Das Vergessen der Klausel kann ähnlich fatale Folgen haben wie ein UPDATE ohne WHERE. Beim Löschen wird dadurch die Tabelle vollständig geleert – in beiden Fällen sind die Daten verloren. Die Optionen ORDER BY und LIMIT werden genutzt, um Reihenfolge und Anzahl der zu löschenden Datensätze festzulegen.[127]

ALTER: Mit dem Befehl ALTER kann man die Struktur von Datenbanken und Tabellen verändern. Für Tabellen lassen sich mehr Modifikationen durchführen als für Datenbanken. Im letzteren Fall beschränken sich die Möglichkeiten des ALTER-Befehls auf den verwendeten Zeichensatz und die Sortierung, wie an folgender Syntax ersichtlich:[128]

```
ALTER DATABASE [dbName]
[ [DEFAULT] CHARACTER SET zeichensatz
| [DEFAULT] COLLATE sortierung ]
```
[129]

Verzichtet man auf die Angabe des Datenbanknamens, bezieht sich die Anweisung auf die aktuelle Standarddatenbank. Mehr Möglichkeiten für Modifikationen gibt es, wie bereits

[124] Reimers und Thies 2012, S. 222.
[125] vgl. Reimers und Thies 2012, S. 222.
[126] Reimers und Thies 2012, S. 223.
[127] vgl. Reimers und Thies 2012, S. 223.
[128] vgl. Reimers und Thies 2012, S. 224.
[129] Reimers und Thies 2012, S. 224.

angesprochen, bei Tabellen. So können beispielsweise Primärschlüssel (PRIMARY KEY) und Eindeutigkeitsbedingungen (UNIQUE) angelegt und gelöscht werden und des Weiteren lässt sich auch die Tabelle mit RENAME umbenennen.[130] Die Syntax der soeben behandelten Befehle sieht wie folgt aus:

```
ALTER TABLE tabelle ADD PRIMARY KEY (attrName2);
ALTER TABLE tabelle DROP PRIMARY KEY;
ALTER TABLE tabelle ADD UNIQUE index1 (attrName2);
ALTER TABLE tabelle RENAME tabelle2;[131]
```

DROP: Mit dem Befehl DROP {DATABASE | TABLE} werden Strukturen nachhaltig gelöscht. Bevor eine Datenbank gelöscht wird, sollten alle Tabellen entfernt werden (Stichwort „saubere Programmierung“):

```
DROP [TEMPORARY] TABLE [IF EXISTS]
tblName [, tblName ...][132]
```

Das optionale IF EXISTS erfüllt eine ähnliche Aufgabe wie das IGNORE – das Löschen einer nicht existierenden Tabelle führt zu keinem Fehler, wenn IF EXISTS im Kommando enthalten ist.[133]

Die Syntax zum Löschen von Datenbanken sieht schlussendlich wie folgt aus:

```
DROP DATABASE [IF EXISTS] dbName[134]
```

Datentypen

MySQL stellt insgesamt 27 Datentypen zur Verfügung, welche sich in folgende vier Gruppen einteilen lassen: 1. Numerische Typen (ganzzahlig und nicht ganzzahlig), 2. Zeichenketten (binär und nicht binär), 3. Datums- und Zeittypen und 4. Mengentypen.[135]

Die Typen einer Gruppe unterscheiden sich prinzipiell lediglich in ihrer Größe bzw. ihren Wertebereichen (ausgenommen sind davon die Mengentypen). Für numerische Typen kann man den Anzeigebereich bestimmen, indem man bei der Definition die Länge in runden Klammern hinter dem Datentyp angibt. Werden Werte verschiedener Datentypen miteinander verglichen, so werden diese von MySQL automatisch konvertiert (diese Konvertierung ist jedoch ausschließlich für die Zeit der Abfrageausführung gültig). Manuelle Typkonvertierung erreicht man mit der Funktion CAST(ausdr AS typ).[136]

Als Typ kann aus folgenden acht Alternativen gewählt werden: SIGNED (vorzeichenbehafteter Integerwert), UNSIGNED (vorzeichenloser Integerwert), DECIMAL (numerischer Fließkomma-

[130] vgl. Reimers und Thies 2012, S. 226.
[131] Reimers und Thies 2012, S. 226.
[132] Reimers und Thies 2012, S. 226.
[133] vgl. Reimers und Thies 2012, S. 226.
[134] Reimers und Thies 2012, S. 227.
[135] vgl. Reimers und Thies 2012, S. 227.
[136] vgl. Reimers und Thies 2012, S. 227-228.

wert), CHAR (Zeichenkette), BINARY (binäre Zeichenkette; binäre Vergleiche unterscheiden zwischen Groß- und Kleinschreibung), DATE (Datum im Format YYYY-MM-DD), DATETIME (Datum und Zeit im Format YYYY-MM-DD HH:MM:SS) und TIME (Zeit im Format HH:MM:SS).[137]

Numerische Datentypen: Es erfolgt eine weitere Aufteilung in ganzzahlige und nicht ganzzahlige Typen, welche jeweils in den folgenden beiden Abbildungen (Abb. 15 und Abb. 16) ersichtlich sind:

Typgruppe	Datentyp	Wertebereich (Größe)
ganzzahlige numerische Typen	TINYINT(3)	–128 bis 127 (2^8)
	SMALLINT(5)	–32.768 bis 32.767 (2^{16})
	MEDIUMINT(8)	–8.388.608 bis 8.388.607 (2^{24})
	INT(10)	–2.147.483.648 bis 2.147.483.647 (2^{32})
	BIGINT(20)	–9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807 (2^{64})

Abbildung 15 Ganzzahlige numerische Typen

Typgruppe	Datentyp	Wertebereich
nicht ganzzahlige numerische Typen	FLOAT[(A,D)]	$-3.402823466*10^{38}$ bis $-1.175494351*10^{-38}$, 0 und von $1.175494351*10^{-38}$ bis $3.402823466*10^{38}$; A ist die Anzeigegröße, D steht für die Anzahl Nachkommastellen.
	DOUBLE [(A,D)]	$-1.7976931348623157*10^{308}$ bis $-2.2250738585072014*10^{-308}$, 0 und von $2.2250738585072014*10^{-308}$ bis $1.7976931348623157*10^{308}$; A und D entsprechen den Angaben bei FLOAT.
	DEC[(A[,D])]	Der Maximalwert ist identisch mit DOUBLE; A steht für die gesamte Anzahl an Stellen, D bezeichnet die Nachkommastellen.
	DECIMAL [(A[,D])]	Der Maximalwert ist identisch mit DOUBLE; A steht für die gesamte Anzahl an Stellen, D bezeichnet die Nachkommastellen.

Abbildung 16 Nicht ganzzahlige numerische Typen

Zeichenketten: Diese werden ebenfalls aufgeteilt – in binäre und nicht binäre Zeichenketten. MySQL definiert sechs nicht binäre Texttypen unterschiedlicher Länge, welche in Abb. 17 dargestellt werden. Bei Sortierungen und Vergleichen wird bei diesen nicht zwischen Groß- und Kleinschreibung differenziert – ein Kennzeichen, mit dem sie sich signifikant von den Binärtypen (siehe Abb. 18) abheben.

[137] vgl. Reimers und Thies 2012, S. 228.

Typgruppe	Datentyp	Länge
nicht binäre Texttypen	CHAR(N)	N zwischen 0 und 255 Zeichen
	VARCHAR(N)	N zwischen 0 und 65.532 Zeichen, abhängig vom Zeichensatz, eigentlich 65.535 Zeichen
	TINYTEXT	255
	TEXT	65.535
	MEDIUMTEXT	16.777.215
	LONGTEXT	4.294.967.295

Abbildung 17 Nicht binäre Texttypen

Typgruppe	Datentyp	Länge
Binärtypen	TINYBLOB	255
	BLOB	65.535
	MEDIUMBLOB	16.777.215
	LONGBLOB	4.294.967.295
	BINARY	Synonym für CHAR BINARY
	VARBINARY	Synonym für VARCHAR BINARY

Abbildung 18 Binärtypen

Datums- und Zeittypen: Das DBMS sieht – anders als in PHP – gleich mehrere Datentypen vor, die sich speziell auf zeitbezogene Daten ausrichten (siehe Abb. 19). So ist man nicht darauf beschränkt, auf einen Integer auszuweichen, um Unix-Zeitstempel darzustellen. Dementsprechend vielfältig sind auch die Funktionen, mit denen sich Zeitdaten manipulieren lassen.[138]

Typgruppe	Datentypen	Beispiel
Datum- und Zeittypen	DATETIME	2010-02-28 23:59:59
	DATE	2010-02-28
	TIMESTAMP	20100228235959
	TIME	23:59:59
	YEAR	2010

Abbildung 19 Datums- und Zeittypen

Mengentypen: Bei allen bisher beschriebenen Attributen wurde deren Wert durch den Datentyp größenmäßig eingeschränkt – bei numerischen Typen ist dies die maximal speicherbare Zahl, bei Texttypen die Länge. Mengentypen (siehe Abb. 20) schränken ihre Werte auf eine andere Weise ein: Ein Attribut kann einen oder mehrere Werte aus einer vordefinierten Liste an Alternativen annehmen.

Typgruppe	Datentypen	maximale Alternativenanzahl
Mengentypen	ENUM	65.535 Elemente
	SET	64 Elemente

Abbildung 20 Mengentypen

[138] vgl. Reimers und Thies 2012, S. 236.

Ein Attribut vom Typ `ENUM` wird mit einer Liste an Alternativen wie folgt definiert:

```
CREATE TABLE mengen(enumeration ENUM('wert1','wert2','wert3',...));
```
[139]

Die Definition eines Attributs vom Typ `SET` erfolgt analog zu `ENUM` und lautet:

```
ALTER TABLE mengen ADD sets SET('wert1','wert2','wert3',...);
```
[140]

3.3 Webserver

Funktionsweise eines Webservers

Das WWW (World Wide Web) besteht im Wesentlichen aus einer großen Anzahl von Servern – darunter befinden sich Dateiserver, Datenbankserver und Webserver. Über den Zugriff auf einen Webserver erreicht man HTML-Webseiten, Multimediadaten und Download-Daten. Der Client, mit dem man Dateien vom Webserver anfordert, ist in der Regel ein Webbrowser. Dieser übernimmt zum einen die Kommunikation mit dem Server, zum anderen stellt er die zurückgeschickten Webseiten auch gleich auf dem Bildschirm des Clientrechners dar. Browser laden Webseiten oder andere angebotene Dateien herunter, wenn man eine gezielte Anfrage danach an einen Server im WWW schickt. Dies macht man beispielsweise, wenn man in der Adresszeile des Browsers eine URL eintippt oder wenn man innerhalb einer Webseite einen (Hyper-)Link anklickt. Diese Aktion geschieht clientseitig – das bedeutet auf dem eigenen Computer. Die Anfrage wird über das Internet an den entsprechenden Server weitergeleitet und dort dann bearbeitet. Webserver warten prinzipiell permanent auf Anfragen, um Dateien an den Client zurückzuschicken. [141] Dieser Nachrichtenaustausch zwischen Client und Server lässt sich schematisch wie folgt darstellen (siehe Abb. 21):

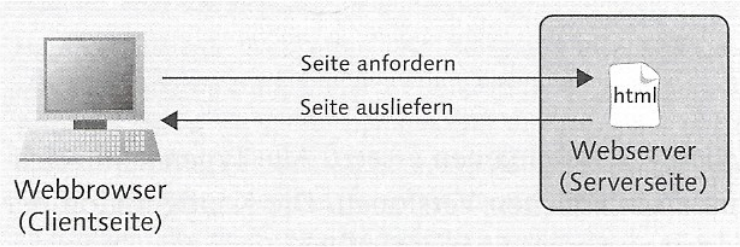


Abbildung 21 Kommunikation zwischen Client und Server

Bearbeitung einer statischen Webseite

Nun soll der Ablauf einer Kommunikation zwischen Client und Server erklärt werden – als Beispiel soll eine Anfrage nach einer Webseite, also einem HTML-Dokument, dienen. Der

[139] Reimers und Thies 2012, S. 241.
[140] Reimers und Thies 2012, S. 241.
[141] vgl. Reimers und Thies 2012, S. 26-27.

Anwender auf der Clientseite gibt in der Adresszeile seines Browsers eine URL ein, woraus sich beim Client eine HTTP-Anfrage entwickelt, die der Browser über das Internet an den entsprechenden Server schickt. Beide Informationen (also die gewünschte Datei und die Adresse des Webservers, von welchem man die Antwort erwartet) befinden sich in der URL. Trifft die Anforderung beim Server ein, lädt dieser das HTML-Dokument von seiner Festplatte und schickt es an den Client, also an den Browser, zurück. Im nächsten Schritt werden alle dazugehörigen Dateien vom Webserver geholt, wie z.B. Bilder oder externe JavaScript- oder CSS-Dateien. All diese Dateien sind statisch, was bedeutet, dass sie vom Webserver nicht bearbeitet werden, sondern genau so an den Client geschickt werden, wie sie auf der Festplatte gespeichert sind. Ob der Webserver eine Bearbeitung durchführen soll, entscheidet er anhand der Dateierweiterung (also z.B. „`.html`“, „`.css`“, „`.js`“) und nicht anhand des Inhalts der Datei. Damit ist dann für eine statische HTML-Seite die Arbeit auf Serverseite erledigt. Sind alle benötigten Dateien beim Browser angekommen, werden sie dort clientseitig verarbeitet: Die Bilder werden in das HTML-Dokument eingefügt, die JavaScript-Anweisungen ausgeführt und die Stylesheets angewendet. Zu dem Zeitpunkt, an dem die Webseite dann vollständig geladen ist, wird die Anfrage und zugleich auch die Verbindung zwischen Server und Client beendet.[142]

Bearbeitung einer dynamischen Webseite

Bei sogenannten dynamischen Webseiten, welche über PHP oder sonstige serverseitige Skriptsprachen erstellt werden, verhält es sich allerdings anders. Hierbei werden die Dateien von dem sogenannten Parser verarbeitet, bevor das Ergebnis der Verarbeitung als Webseite an den Browser des Clients zurückgeschickt wird. Der Parser ist beim Webserver als Modul registriert (siehe Abb. 22). Alle Dateitypen, die von einem bestimmten Parser verarbeitet werden sollen, sind ebenfalls in der Konfiguration des Webservers festgelegt – typisch für die Anwendung des PHP-Parsers sind dies die folgenden Dateitypen bzw. Dateiendungen: „`.php`“, „`.php3`“, „`.php4`“ und „`.php5`“. Alle diese Typen kennzeichnen PHP-Code, jedoch in unterschiedlichen Versionen.[143]

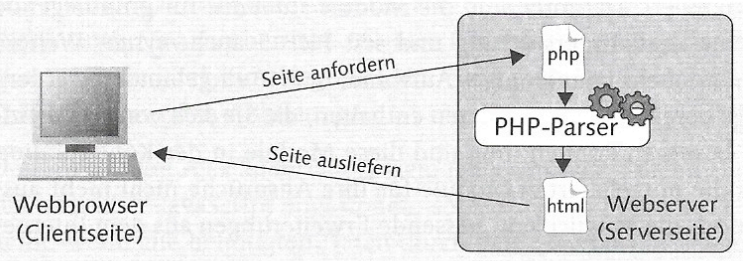


Abbildung 22 PHP Parser

[142] vgl. Reimers und Thies 2012, S. 27-28.
[143] vgl. Reimers und Thies 2012, S. 28.

Prinzip der Blackbox

Wichtig ist, dass sich für den Client ein Aufruf einer HTML-Webseite nicht von dem einer PHP-Webseite unterscheidet. Die Ausführung einer HTTP-Anfrage und aller ihrer Teilschritte erfolgt nach dem Blackbox-Prinzip, was bedeutet, dass der Besucher einer Webseite keinerlei Kenntnis darüber hat, ob die angezeigten Daten in einer Datei auf dem Server stehen, in einer externen Datei ausgelagert oder in einer Datenbank gespeichert sind. Bei umfangreichen Seiten kann es lediglich eine längere Ladezeit geben, wenn der HTML-Code von einem PHP-Parser erzeugt werden muss.[144]

Aufgabe des Parsers

Vonseiten des Servers ergeben sich geringfügige Änderungen im Ablauf. Sobald der Webserver die Anfrage empfangen und die PHP-Datei vom Speichermedium geladen hat, setzt der Parser ein, wobei hier Anweisung für Anweisung in dem Skript abgearbeitet wird. Dabei können u.a. temporäre Dateien auf der Festplatte des Servers angelegt, externe Quellen, wie z.B. Datenbanken oder Dateien ausgelesen, weitere Anfragen an entfernte Server gestellt oder sogar E-Mails versendet werden. Schlussendlich wird dann in den meisten Fällen eine HTML-Ausgabe erzeugt (siehe „3.1.3 Strukturen einer PHP-Seite“).[145]

Module

Die Funktionalitäten sind auch bei PHP als Module angelegt – jedes Modul erweitert den PHP-Kern um eine gewisse Funktionalität (so lässt sich z.B. über das Modul *mysqli* eine Verbindung zu einem MySQL-Server herstellen). Über Module kann man die Fähigkeiten von PHP in viele Richtungen ausdehnen. Ein wichtiges wäre beispielsweise MySQL, welches ja bereits im vorigen Kapitel („3.2.3 MySQL und SQL“) ausführlich behandelt wurde. Die Rolle von MySQL oder anderen Datenbankmanagementsystemen ist durchaus mit den Funktionen eines Webservers zu vergleichen: So wartet ein Datenbankserver nämlich passiv auf eine an ihn gestellte Anfrage und sobald eine Anforderung eintrifft, arbeitet der Server sie ab und schickt das Ergebnis zurück an die Quelle der Anfrage.[146]

Relationale Datenbankmanagementsysteme wie MySQL liefern Ergebnisse in Tabellenform – dabei handelt es sich also um Rohdaten, die beim Empfänger weiterverarbeitet oder zumindest noch in eine darstellbare Form gebracht werden müssen. Anders ist dies bei dem Anfrageergebnis von einem Webserver, das ohne Überarbeitung beim Client dargestellt wird.

[144] vgl. Reimers und Thies 2012, S. 29.
[145] vgl. Reimers und Thies 2012, S. 29.
[146] vgl. Reimers und Thies 2012, S. 30.

Die Aufgabe der Weiterverarbeitung übernimmt dann PHP.[147] An folgender Abbildung (Abb. 23) wird dies veranschaulicht:

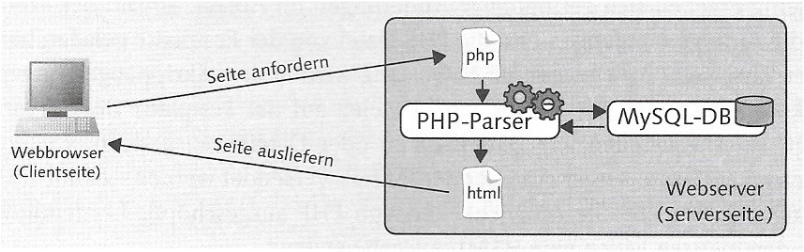


Abbildung 23 Blackbox-Prinzip

Des Weiteren ist es mit PHP möglich, Daten in eine MySQL-Datenbank einzutragen. Auch das Speichern von Daten auf einer Webseite geschieht nach dem Blackbox-Prinzip: Das heißt, der Anwender weiß nicht, ob die Werte, die er eben in ein Webformular eingegeben hat, in einer Datei auf der Festplatte des Servers oder in einer Datenbank hinterlegt werden. Dementsprechend sind für den Besucher eine Webseite die Situationen aus den Abbildungen 21, 22 und 23 auch vollkommen identisch.[148]

3.4 Kryptographie

3.4.1 Allgemeines zu Kryptosystemen

Heutzutage ist eine verschlüsselte Übermittlung von Nachrichten nicht mehr nur für Militärs oder Geheimagenten von Wichtigkeit, sondern nahezu für alle Daten, die übertragen werden.[149] Beispiele hierfür wären im privaten Bereich das Online-Banking oder seit Neuestem (Stand April 2016) die Ende-zu-Ende-Verschlüsselung bei WhatsApp.[150]

Was genau versteht man nun aber unter Kryptosystemen? Die Antwort auf diese Frage lautet wie folgt: Elemente, die für eine sichere Kommunikation zwischen zwei Endpunkten erforderlich sind, werden in ihrer Gesamtheit Kryptosystem genannt. Es handelt sich dabei um ein Ver- und Entschlüsselungssystem.[151]

Die folgende Abbildung veranschaulicht die prinzipielle Struktur eines typischen Kryptosystems:

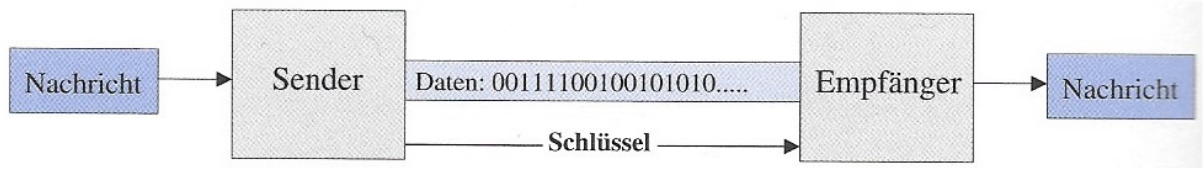


Abbildung 24 Ein typisches Ver- und Entschlüsselungssystem

[147] vgl. Reimers und Thies 2012, S. 30.
[148] vgl. Reimers und Thies 2012, S. 31.
[149] vgl. Herold, Lurz und Wohlrab, S. 772.
[150] https://www.whatsapp.com/faq/de/general/28030015, aufgerufen am 3.5.2016.
[151] vgl. Herold, Lurz und Wohlrab, S. 772.

Nun zur Beschreibung des Vorgangs, der anhand der Abb. 24 ersichtlich ist: Der Sender sendet eine Nachricht (auch Klartext genannt) an den Empfänger, indem er den Klartext in einen so genannten Chriffretext umwandelt, wobei er einen Verschlüsselungsalgorithmus und gewisse Schlüsselparameter verwendet. Um die Botschaft zu lesen, muss der Empfänger den hierzu passenden Entschlüsselungsalgorithmus und die gleichen Schlüsselparameter verwenden, um den Chiffretext in den Klartext zurückwandeln und lesen zu können. Allgemein gilt Folgendes: Ein Kryptosystem ist umso sicherer – zugleich jedoch seine Benutzung umso komplizierter – je mehr Schlüsselparameter vorhanden sind.[152]

3.4.2 Einfache Verschlüsselungsmethoden

Cäsar-Chiffre

Die Cäsar-Verschlüsselung zählt zu den einfachsten und ältesten Verschlüsselungsmethoden. Wie sie funktioniert, wird nun erklärt: Falls ein Buchstabe im Klartext der `n`-te Buchstabe aus dem Alphabet ist, so ersetzt man ihn durch den `(n+k)`-ten Buchstaben aus dem Alphabet, wobei `k` eine feste Zahl ist. Cäsar verwendete übrigens `k = 3`.[153]

Folgendes Beispiel veranschaulicht die Verschlüsselung der Botschaft `NACHRICHT` unter Verwendung der Cäsar-Chiffre mit `k = 3`:

Klartext:	`NACHRICHT`
Chiffretext:	`QDFKULFKW`

Einschränkend muss angemerkt werden, dass diese Methode als äußerst unzuverlässig gilt, da zum Entschlüsseln nur der Wert `k` erraten werden muss. Probiert man jede der 26 in Frage kommenden Möglichkeiten aus, so kann man sich sicher sein, dass man die Botschaft entschlüsseln kann.[154]

Chiffre mit eigener Zuordnungstabelle

Die folgende Vorgehensweise ist eine bessere Methode: Für jeden Buchstaben im Text wird in einer eigenen Tabelle angegeben, welcher Buchstabe im Chiffretext zu verwenden ist.[155] Wenn z.B. folgende Zuordnung vorgegeben ist:

```
_ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A S D F G H J K L Y X C V B M N Q W E R Z T U I O P _
```

dann wird eine Botschaft wie folgt verschlüsselt:

Klartext:	`MEIN_GEHEIMTEXT`
Chiffretext:	`BHYMAKHLHYBZHOZ`

[152] vgl. Herold, Lurz und Wohlrab, S. 772.
[153] vgl. Herold, Lurz und Wohlrab, S. 772-773.
[154] vgl. Herold, Lurz und Wohlrab, S. 773.
[155] vgl. Herold, Lurz und Wohlrab, S. 773.

Unschwer ist zu erkennen, dass diese Art der Verschlüsselung schwieriger zu entschlüsseln ist, wenn die entsprechende Zuordnungstabelle nicht bekannt ist, als dies bei der einfachen Cäsar-Verschlüsselung der Fall war. In diesem Fall müsste man nämlich ungefähr 27! (das entspricht 10^{28}) Tabellen ausprobieren, um eine sichere Entschlüsselung zu gewährleisten.[156] Durch Untersuchungen der Häufigkeiten von Buchstaben und Buchstabenkombinationen kann ein sogenannter Kryptoanalytiker jedoch eine Chiffre durch einfaches Ersetzen sehr leicht entschlüsseln.[157] Dementsprechend handelt es sich hierbei zwar um ein sichereres Verfahren als die Cäsar-Verschlüsselung, aber es ist in dieser Hinsicht dennoch nicht das „Gelbe vom Ei".

3.4.3 Vigenère-Verschlüsselung

Die Verwendung von mehr als nur einer Tabelle ermöglicht es, den Weg zur Entschlüsselung zu erschweren. Ein Beispiel hierfür ist eine Verallgemeinerung der Cäsar-Chiffre, welche Vigenère-Chiffre genannt wird: Dabei wird ein kurzer, sich wiederholender Schlüssel benutzt, um den Wert von `k` für jeden Buchstaben neu zu bestimmen. Bei jedem Schritt wird der Index des Buchstabens im Schlüssel zum Index des Buchstabens neu addiert, um den Index des Buchstabens im Chiffretext zu bestimmen.[158] So lässt sich mit dem Schlüssel `CBA` der Klartext `NACHRICHTX` wie folgt verschlüsseln:

Schlüssel:	`CBACBACBAC`
Klartext:	`NACHRICHTX`
Chiffretext:	`QCDKTJFJUA`

`N` im Klartext ist der 14. Buchstabe im Alphabet und das `C` im Schlüssel der 3. Buchstabe. Daraus ergibt sich für den Chiffretext der 17. Buchstabe (aus 14+3), also Buchstabe `Q`. Nimmt man das `X` aus dem Klartext und das `C` aus dem Schlüssel, so erhält man bei der Addition von `X` (24. Buchstabe) und `C` (3. Buchstabe) 27. Da im Alphabet aber nur 26 Buchstaben existieren, muss eine Modulo-26 Rechenoperation durchgeführt werden, welche schließlich das Ergebnis 1 liefert und was bedeutet, dass der 1. Buchstabe, also `A`, für den Chiffretext genommen wird.[159]

Je länger der Schlüssel bei der Vigenère-Verschlüsselung ist, umso schwerer ist der Chiffretext für Fremde zu entschlüsseln. Wenn der Schlüssel genauso lang wie der Klartext ist, spricht man auch von der so genannten Vernam-Chiffre. Dies ist das einzige nachweisbar sichere Kryptosystem: Da jeder Buchstabe im Schlüssel nur einmal verwendet wird, hat ein

[156] vgl. Herold, Lurz und Wohlrab, S. 773.
[157] vgl. Herold, Lurz und Wohlrab, S. 773.
[158] vgl. Herold, Lurz und Wohlrab, S. 773-774.
[159] vgl. Herold, Lurz und Wohlrab, S. 774.

Außenstehender keine andere Möglichkeit, als für jede Position der Nachricht jeden möglichen Schlüsselbuchstaben auszuprobieren, was man als hoffnungsloses Unterfangen bezeichnen kann.[160]

Hier stellt sich nun aber die Frage, warum dieses scheinbar perfekte System nur so selten eingesetzt wird. Die Antwort darauf lautet wie folgt: Das Problem besteht in der Übermittlung des Schlüssels, der genauso lang wie der Klartext sein muss. Wenn man den Schlüssel auf demselben Weg wie den Klartext übermittelt, so ist – nämlich wegen der Länge des Schlüssels – die Chance gelesen zu werden so groß wie bei einer unverschlüsselten Übermittlung des Klartextes. Versucht man, den Schlüssel auf einem anderen Weg zu übermitteln (wie z.B. durch die Post), so hat man das Risiko nur auf diesen anderen Übermittlungsweg verlagert, das Problem also leider nicht gelöst.[161]

3.4.4 Verschlüsselung mittels Zufallsfolgen

Bei einem binären Klartext bietet sich an, auch einen binären Schlüssel zu verwenden. In diesem Fall entsteht der Chiffretext, indem man die Bits des Schlüssels mittels `XOR` (es handelt sich hierbei um das „exklusive Oder“) mit den Bits des Klartextes verknüpft. Eine nützliche Eigenschaft dieser Methode ist, dass man sowohl für die Verschlüsselung als auch für die Entschlüsselung die `XOR`-Verknüpfung verwenden kann.[162] Dies ist an folgenden Beispielen ersichtlich:

Verschlüsselung:

Schlüssel:	`11110000`
Klartext:	`01101101` (`XOR`)
Chiffretext:	`10011101`

Entschlüsselung:

Schlüssel:	`11110000`
Chiffretext:	`10011101` (`XOR`)
Chiffre:	`01101101`

Weiters funktioniert es auch, mittels `XOR` den Schlüssel zu ermitteln, wenn man den Klartext und den Chiffretext kennt:

Finden des Schlüssels aus Klartext und Chiffretext:

Klartext:	`01101101`
Chiffretext:	`10011101`
Schlüssel:	`11110000`

[160] vgl. Herold, Lurz und Wohlrab, S. 774.
[161] vgl. Herold, Lurz und Wohlrab, S. 774.
[162] vgl. Herold, Lurz und Wohlrab, S. 774.

Wo wären Einsatzmöglichkeiten für diese Art der Verschlüsselung denkbar? Als Beispiel könnte man hier private Fernsehgesellschaften (Pay-TV) anführen, welche dieses Verschlüsselungsverfahren anwenden, um sich vor Schwarzsehern zu schützen.[163]

3.4.5 Kryptosysteme mit öffentlichen Schlüsseln

Eigenschaften von Public-Key-Systemen

Bei den bisher erörterten Chiffriersystemen gelten immer die folgenden zwei Aspekte:

1. Wer verschlüsseln kann, kann auch entschlüsseln.
2. Je zwei Partner müssen einen gemeinsamen geheimen Schlüssel austauschen.[164]

Während man die erste Eigenschaft als Vorteil ansieht, muss die zweite Eigenschaft als Nachteil angesehen werden. Die soeben besprochenen Verfahren zeichnen sich nun aber dadurch aus, dass sie sich von der ersten Eigenschaften so weit wie möglich entfernen und die zweite Eigenschaft überhaupt nicht ausweisen, was bedeutet, dass der Schlüssel nicht mehr geheim, sondern jeder Person zugänglich ist.[165]

Die Idee von Kryptosystemen mit öffentlichen Schlüsseln (sogenannten Public-Key-Systemen) besteht in der Verwendung eines „Telefonbuchs mit Schlüsseln“ für die Verschlüsselung. Der Schlüssel von jedem ist für die Verschlüsselung (mit `P` bezeichnet) öffentlich bekannt. Der Schlüssel einer Person könnte z.B. neben ihrer Nummer mit Telefonbuch angegeben sein. Jeder besitzt außerdem einen privaten Schlüssel zur Entschlüsselung (mit `S` bezeichnet), den sonst niemand kennt.[166]

Wie wird nun eine Botschaft übermittelt, wie läuft dieser Vorgang ab? Dazu sucht der Absender den öffentlichen Schlüssel `P` des Empfängers heraus, verschlüsselt seine Botschaft damit und übermittelt dann die chiffrierte Botschaft `C = P(M)`. Der Empfänger verwendet dann seinen privaten Schlüssel `S` für das Entschlüsseln der chiffrierten Botschaft und erhält somit die originale Botschaft `M = S(C) = S(P(M))`.[167]

Damit dieses System funktioniert, müssen jedoch folgende Bedingungen erfüllt sein:

- `S(P(M)) = M` für jede Botschaft `M`.
- Alle Paare `(S,P)` sind verschieden.
- Das Finden des privaten Schlüssels `S` bei Kenntnis von `P` ist nahezu unmöglich.
- Das Entschlüsseln der Botschaft `M` ohne Kenntnis des Schlüssels `S` ist nahezu unmöglich.

---

[163] vgl. Herold, Lurz und Wohlrab, S. 775.
[164] Herold, Lurz und Wohlrab, S. 776.
[165] vgl. Herold, Lurz und Wohlrab, S. 776.
[166] vgl. Herold, Lurz und Wohlrab, S. 776.
[167] vgl. Herold, Lurz und Wohlrab, S. 776.

- Sowohl `S` als auch `P` lassen sich leicht berechnen. Diese Bedingung ist für die praktische Verwendbarkeit des Systems unverzichtbar.[168]

Solche Public-Key-Systeme wurden 1976 von Diffie und Hellman vorgeschlagen, was damals eine revolutionäre Idee für die Kryptographie bedeutete, jedoch konnten sie keine Methode angeben, die alle diese soeben genannten Bedingungen erfüllt hätte. Eine derartige Methode wurde bald danach von Rivest, Shamir und Adleman (Abkürzung RSA) gefunden. Ihr Verfahren wurde als RSA-Kryptosystem mit öffentlichen Schlüsseln bekannt.[169]

Der RSA-Algorithmus kann leider aufgrund seiner Komplexität und des Umfangs in dieser Arbeit keine Ausarbeitung für sich beanspruchen.

[168] Herold, Lurz und Wohlrab, S. 776.
[169] vgl. Herold, Lurz und Wohlrab, S. 776.

4. Entschlüsselung der Zahlenrätsel von Ephesos

In diesem Kapitel geht es um die Beschreibung der konkreten Umsetzung des interdisziplinären Projekts „Entschlüsselung der Zahlenrätsel von Ephesos“.

Zuerst wird erklärt, worum es sich bei den Zahlenrätseln von Ephesos handelt, dann wird der Weg von einer einfachen Umrechnungstabelle über verschiedene Ausnahmen und Restriktionen, die es zu berücksichtigen gab, bis hin zum fertigen, funktionsfähigen Programm dargelegt. Dazwischen mussten noch viele weitere Schritte erledigt werden. So musste beispielsweise die Datenbank vorbereitet und mit speziellen Datensätzen befüllt werden und das Web Interface erstellt und gestaltet werden. Den größten Teil dieses Projekts machte – kaum überraschend – die Erstellung des Programms (also die Programmierung in PHP, sowie Datenmodellierung und -abfragen mit SQL) aus.

All jene soeben genannten Arbeitsschritte werden nun im Folgenden jeweils erörtert und so gut wie möglich veranschaulicht. Die gesamte Code Dokumentation und der konkrete Ablauf des Programms (Beschreibung und Screenshots) und die Ergebnisse, welche bereits mit Hilfe dieses Programms in der Forschung erzielt werden konnten, runden dieses Kapitel schließlich ab.

4.1 Was sind die Zahlenrätsel von Ephesos?

Rätsel haben immer schon eine besondere Faszination auf Menschen ausgeübt – so ist dies auch bei den Zahlenrätseln von Ephesos, auch wenn diese bisher eher unbekannt waren. Seit vielen Jahren ist das Institut für Alte Geschichte (unter der Leitung von Herrn ao. Univ.-Prof. Dr. Taeuber) der Universität Wien in Ephesos auf Ausgrabungen unterwegs und befasst sich mit dessen Erforschung, so auch unter anderem mit den Zahlenrätseln, die zu antiker Zeit als Graffiti an den Hauswänden im Hanghaus 2 gefunden wurden (siehe „6.2 Die Hanghäuser von Ephesos“). Diese beinhalten Zahlencodes, welche vom Autor der Rätsel verwendet wurden, um Namen zu kodieren. Wie diese Zahlencodes aufgebaut sind bzw. wie die Zahlenrätsel funktionieren soll nun folgender Text erläutern, welcher von Herrn ao. Univ.-Prof. Dr. Taeuber verfasst und für das Web Interface zur Verfügung gestellt wurde – ebenfalls wie die Fotos der Zahlenrätsel (siehe Abb. 25 und 26), welche sich auf der nächsten Seite befinden:

> Die Buchstaben des griechischen (wie z.B. auch des hebräischen) Alphabets hatten neben ihrem Laut- auch einen Zahlwert. Das System war in drei Neunergruppen gegliedert, je eine Gruppe repräsentierte die Einer, Zehner und Hunderter. Es begann mit Alpha = 1, Beta = 2 usw. bis Theta = 9; dann setzten die Zehner mit Iota = 10, Kappa = 20 … fort, und schließlich folgten die Hunderter mit Rho = 100, Sigma = 200 etc. Da das klassische griechische Alphabet

nur 24 Buchstaben umfaßte, hat das Zahlsystem drei damals in der Schrift bereits ungebräuchliche Buchstaben weiterverwendet: Stigma (ϛ) = 6, Koppa (Ϙ) = 90 und Sampi (ϡ) = 900. Ab 1000 wurde wieder mit Alpha begonnen, allerdings mit diakritischem Zeichen (im Druck als ͵α wiedergegeben), um es vom einfachen α′ = 1 zu unterscheiden.

Dieses System ermöglichte es, von jedem beliebigen Wort seinen Zahlwert zu berechnen und es damit für Uneingeweihte zu verschlüsseln (isopsephische Rätsel, von *ísos* = gleich und *pséphos* = Stein beim Rechenbrett/Abacus). Dies wird z.B. auch im christlichen Bereich angewandt, wo Qθ = 99 für AMHN steht (A = 1, M = 40, H = 8, N = 50). Besonders gerne wird diese Codierung für erotische Rätsel verwendet, wenn der Name der oder des Geliebten erraten werden soll. Dafür gibt es unter den Graffiti in Ephesos und anderswo zahlreiche Beispiele.

Abbildung 25 Zahlenrätsel von Ephesos (Teil 1)

Abbildung 26 Zahlenrätsel von Ephesos (Teil 2)

Ich habe von Herrn ao. Univ.-Prof. Dr. Taeuber vier bekannte Rätsel übermittelt bekommen, welche im Verlauf der Arbeit gelöst werden konnten. Diese befinden sich nun auf der Webseite. Die folgenden zwei Abbildungen (Abb. 27 und Abb. 28) zeigen die Rätsel inklusive einer Auswahl der berechneten Lösungen.

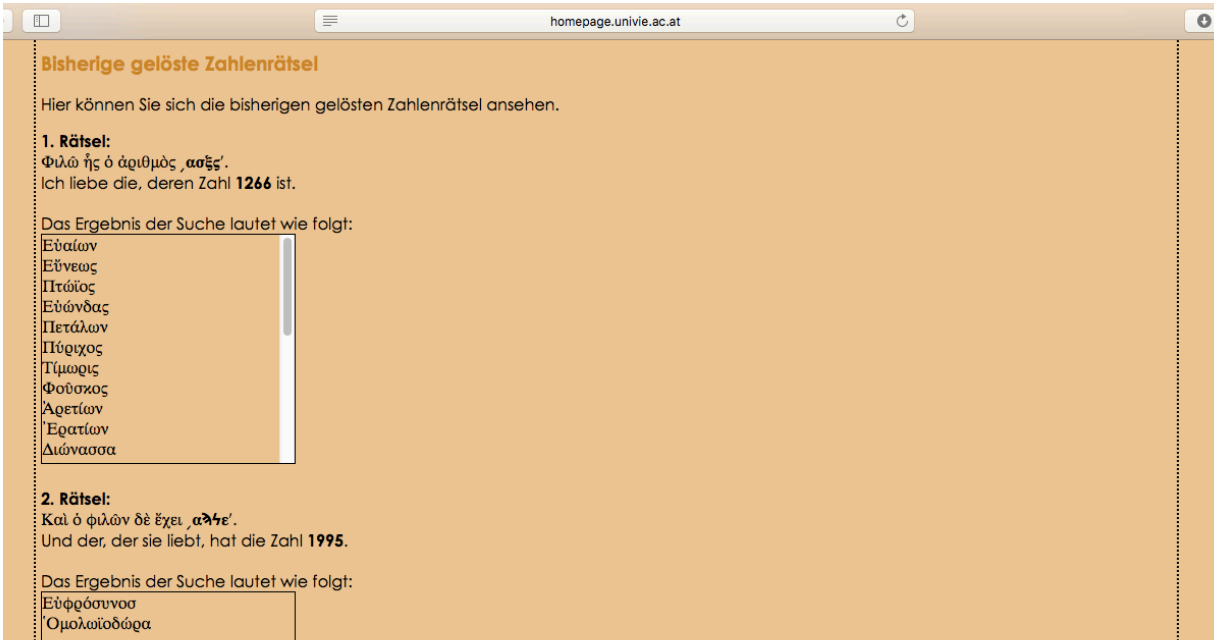


Abbildung 27 Screenshot der gelösten Rätsel (Teil 1)

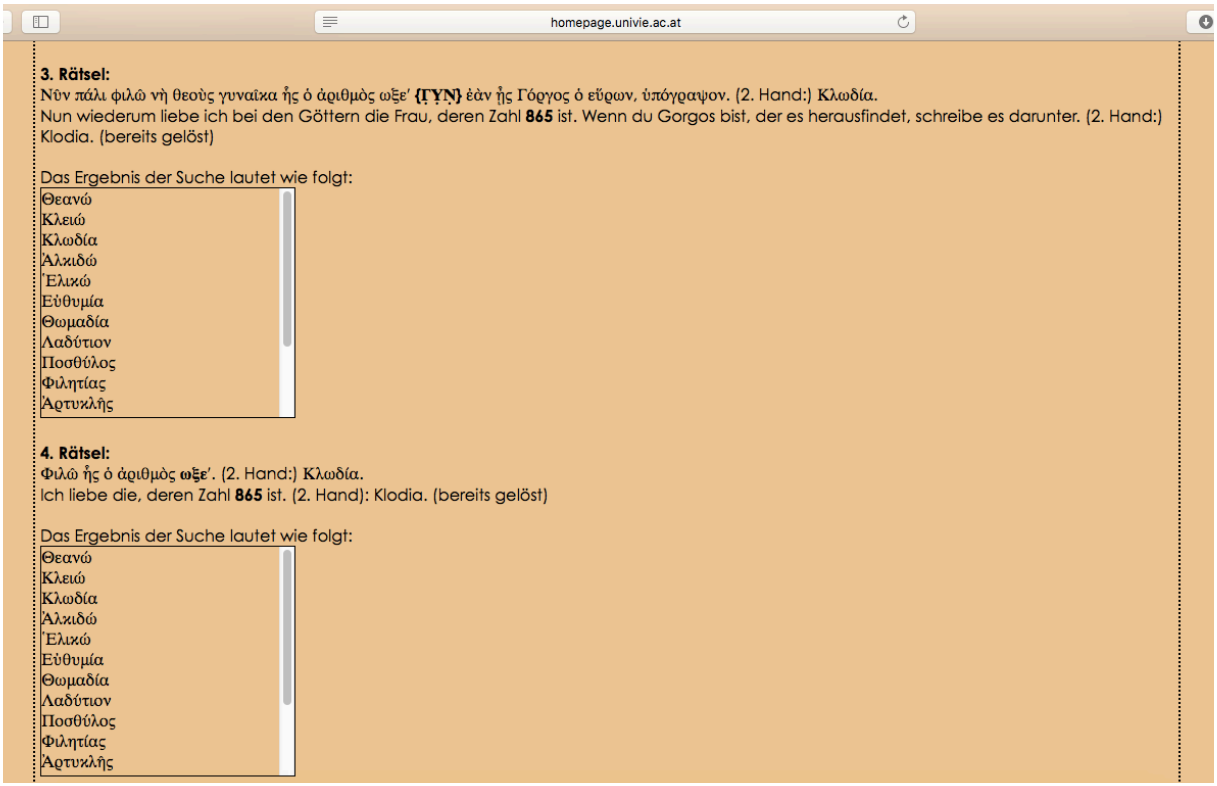


Abbildung 28 Screenshot der gelösten Rätsel (Teil 2)

4.2 Projektanforderungen

Die Projektanforderung war, ein Programm zu erstellen, was es ermöglicht, bestimmte Rätsel (eine Zahl oder ein griechischer Name) mit Hilfe einer Umrechnungstabelle zu lösen (diese Umrechnungstabelle ordnet jedem Buchstaben einen Zahlenwert zu). Die daraus berechneten Ergebnisse sollten aus einer Menge an griechischen Namen, die in der Datenbank gespeichert wurden, ausgewählt werden. Weiters sollte es möglich sein, verschiedene Ausnahmen in die Berechnung miteinzubeziehen (z.B. Vertauschen von Buchstaben). Diese Berechnungen sollten in beide Richtungen funktionieren. Dieses Programm sollte als ein interaktives Web Interface entwickelt werden, welches jederzeit erreichbar sein soll. Auf dieser Webseite sollen auch alle bisher gelösten Rätsel aufgelistet und entschlüsselt werden.

4.3 Umrechnungstabelle

4.3.1 Beschreibung

Die Umrechnungstabelle, welche mir von meinem Betreuer, Herrn ao. Univ.-Prof. Dr. Taeuber, zur Verfügung gestellt wurde, sieht wie folgt aus (Abb. 29):

Aus: A.G.Woodhead, The Study of Greek Inscriptions ²1981

The table of alphabetic numerals which follows omits the marks customarily used in modern texts to draw attention to the fact that numbers are involved. These usually take the form of a mark *above* the line, like an acute accent (′), *following* numbers up to 999, and, in the case of numbers of 1000 and above, a similar mark in addition, *below* the line and *preceding* the numeral. For example, 555 is customarily shown as φνε′, 5555 as ,εφνε′.

α	1	ι	10	ρ	100
β	2	κ	20	σ	200
γ	3	λ	30	τ	300
δ	4	μ	40	υ	400
ε	5	ν	50	φ	500
ϛ (ϝ)	6	ξ	60	χ	600
ζ	7	ο	70	ψ	700
η	8	π	80	ω	800
θ	9	ϙ	90	ϡ	900

Abbildung 29 Umrechnungstabelle

Buchstabe-Zahl-Zuordnung

Wie man erkennen kann, wird jedem Buchstaben des griechischen Alphabets eine bestimmte Zahl zugeordnet – je nach Spalte ist dies entweder eine einstellige, zweistellige oder dreistellige Zahl. Zusätzlich zu den 24 Buchstaben des klassischen griechischen Alphabets

sind noch drei, damals bereits in der Schrift ungebräuchliche Buchstaben, dazugekommen, nämlich Stigma (ϛ) = 6, Koppa (Ϙ) = 90 und Sampi (ϡ) = 900).

4.3.2 Besondere Anforderungen

Umgang mit Akzenten und Spiritus

Wichtig bezüglich der Zuordnung ist, dass sowohl Groß- als auch Kleinbuchstaben eines bestimmten griechischen Buchstabens denselben Zahlenwert zugewiesen bekommen. Des Weiteren sind alle Akzente und Spiritus für die Umrechnung ohne Wert, was bedeutet, dass z.B. α, ἀ, ά und ᾳ (letzteres ist ein sogenanntes „iota subscriptum“, welches unter Vokalen wie α, ι oder ω stehen kann) alle denselben Wert, in diesem Fall 1, haben. Dies war vor allem von großer Bedeutung beim Einfügen der Umrechnungstabelle in die Datenbank, da die Datenbank sehr wohl zwischen diesen Zeichen differenziert und somit beim Vergleichen kein Zahlenwert gefunden werden konnte, wenn nicht alle Varianten und der dazugehörige Wert in der Umrechnungstabelle eingetragen sind (genauere Erklärung siehe Kapitel „4.4.1 Vorbereitung der Datenbank).

Unterscheidung der zwei verschiedenen Sigmas (σ und ϛ)

Eine weitere Problematik lieferte die Unterscheidung zwischen Sigma (σ) und Schlusssigma (ϛ). Für die Lösung der Zahlenrätsel mussten nämlich zwei verschiedene Werte in der Umrechnungstabelle gespeichert werden (doch für die Datenbank sind diese zwei Zeichen identisch und gibt immer nur einen Wert zurück, in dem Fall entweder 6 oder 200). Dieses Problem wurde im Code direkt gelöst (siehe „4.5.2 Code Dokumentation“).

Ausnahmen

Einen nächsten wichtigen Aspekt, den es zu berücksichtigen galt, bildeten die Ausnahmen. Folgende sollten in allen möglichen Kombinationen in die Entschlüsselung miteinbezogen werden können (wenn Checkbox im Programm angekreuzt; ansonsten erfolgt die Berechnung prinzipiell ohne Ausnahmen):

- ϛ = σ (das Schlusssigma erhält denselben Wert wie das „normale“ Sigma, also 200, anstatt 6 wie in der Umrechnungstabelle)
- αι = ε
- ει = ι
- ι = ει
- η = ι
- ω = ο
- ιος = ις
- ιον = ιν
- ου (vor Vokal) = β
- β (vor Vokal) = ου

Wie die Umsetzung dieser Ausnahmen im Code bewerkstelligt wurde, wird in einem späteren Unterkapitel veranschaulicht (siehe „4.5.2 Code Dokumentation“).

Umrechnungen „in beide Richtungen“

Sowohl die Entschlüsselung der Rätsel, indem man eine Zahl eingibt und die entsprechenden Namen aus der Datenbank angezeigt werden, war gefordert, als auch das umgekehrte Prozedere: Hier sollte es möglich sein, einen beliebigen griechischen Namen einzugeben und abzuschicken und als Lösung den jeweiligen berechneten Zahlenwert zu erhalten.

4.4 Datenbank

4.4.1 Vorbereitung der Datenbank

Um die Datenbank vorbereiten und damit arbeiten zu können, mussten zuerst die Login-Daten und alle notwendigen Daten zum Verbinden mit der Datenbank auf den, von der Universität Wien bereitgestellten, Server beschafft werden (siehe Abb. 30, Abb. 31 und Abb. 32). Dies gestaltete sich jedoch nicht schwierig, da alles auf der folgenden Seite vom ZID (Zentraler Informatikdienst) zu finden war: http://zid.univie.ac.at/persoenliche-webseiten/. Zur Administration der Datenbank wurde die Webanwendung phpMyAdmin verwendet.

Abbildung 30 Datenbanken verwalten – ZID

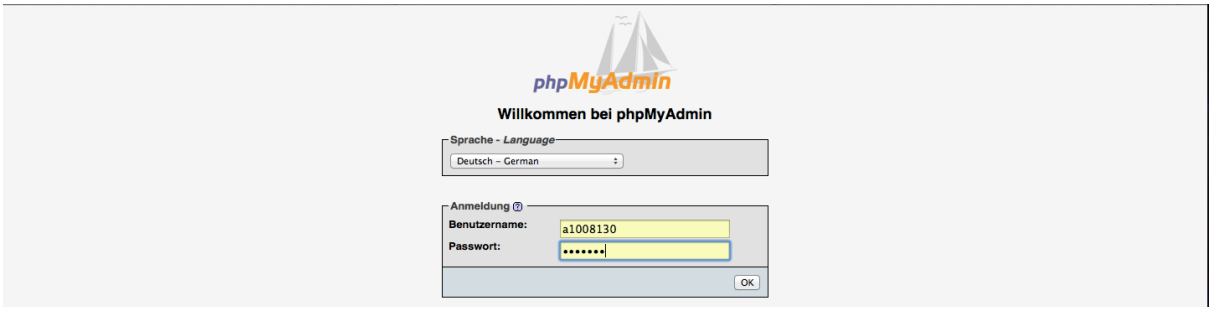


Abbildung 31 Login zu phpMyAdmin

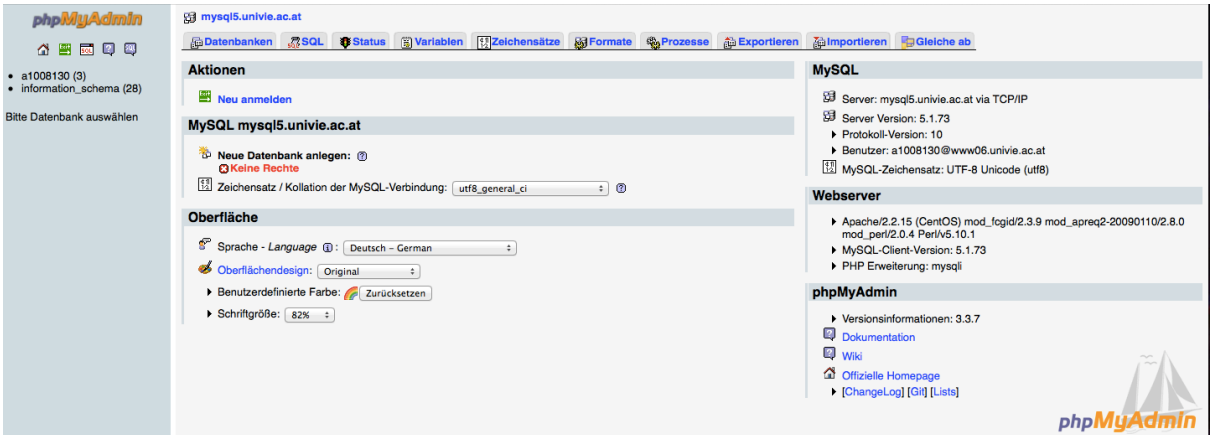


Abbildung 32 Startbildschirm phpMyAdmin

4.4.2 Tabellen

Nun konnte mittels phpMyAdmin die Datenbank angelegt werden und in weiterer Folge die Tabellen, die benötigt wurden, nämlich die folgenden drei (auch an Abb. 33 ersichtlich):

1. tbl_grnamen
2. tbl_raetsel
3. tbl_umrechnung

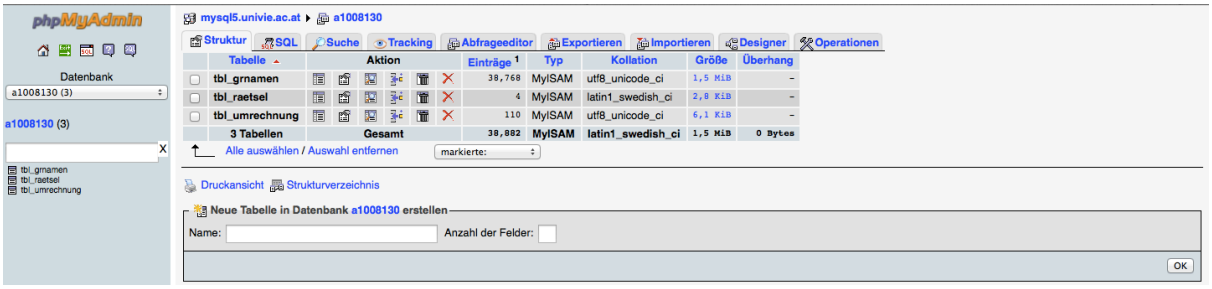


Abbildung 33 Datenbank mit 3 Tabellen

In folgender Abbildung (Abb. 34) sind die Tabellen mit ihren dazugehörigen Attributen dargestellt.

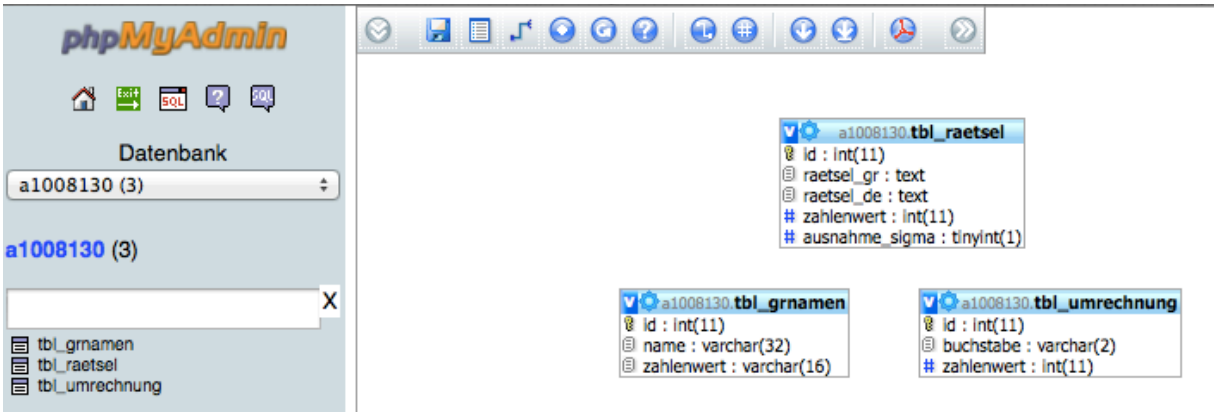


Abbildung 34 Tabellen und ihre Attribute

Nun werden die unterschiedlichen Tabellen genauer beschrieben.

Tabelle mit den griechischen Namen: „tbl_grnamen“

Diese Tabelle enthält die folgenden drei Attribute: „id“, „name“, „zahlenwert“. In diese Tabelle wurden die Daten der von Herrn ao. Univ.-Prof. Dr. Taeuber bereitgestellten csv-Datei mit den griechischen Namen eingespeist (anfangs 39.741 Datensätze). Nach einer Löschung der unvollständigen Namen (wenn z.B. die Endung fehlte oder ein oder mehrere Buchstaben im Wortinneren nicht bekannt waren) betrug die Anzahl der Datensätze dann 38.768. Einen Ausschnitt aus dieser Tabelle zeigt Abb. 35:

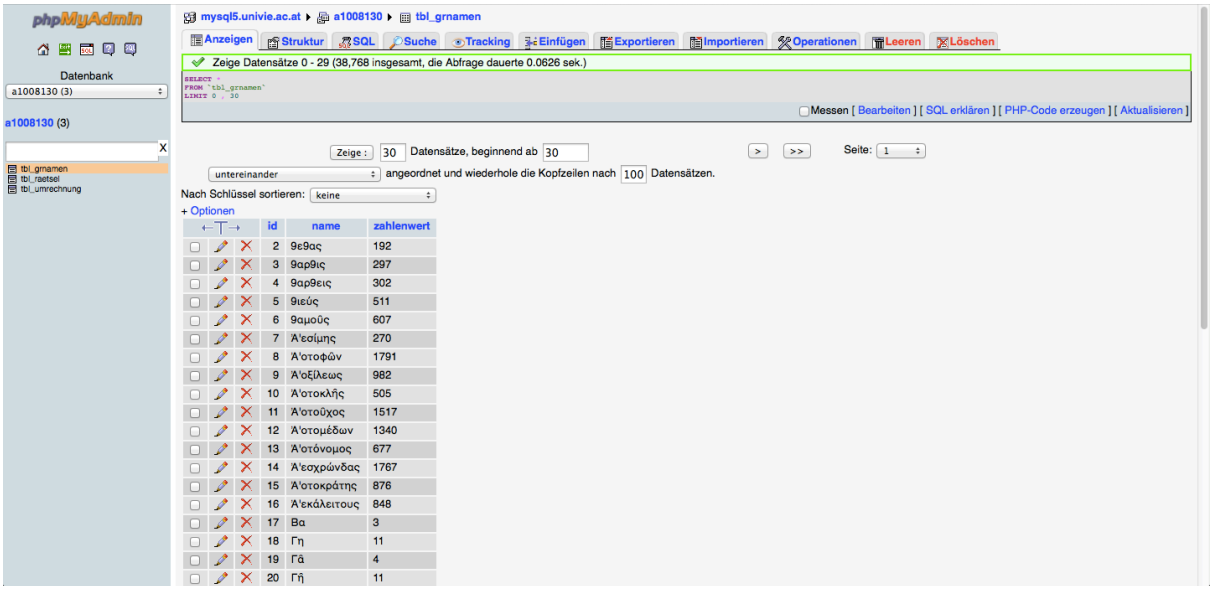


Abbildung 35 Tabelle mit den griechischen Namen (tbl_grnamen)

Wie in Abb. 36 ersichtlich, wurde das Attribut „id“ als Integer angelegt und der Tabelle als Primärschlüssel zugewiesen. Weiters wurde dieses mit „AUTO_INCREMENT“ belegt, was bedeutet, dass man sich beim Anlegen eines neuen Datensatzes um diesen Wert nicht selbst kümmern muss, da dieser automatisch hochgezählt wird.

Das Datenbankattribut „name“ enthält alle, aus der csv-Datei importierten, griechischen Namen. Es wurde als „Varchar“ mit der Länge 32 angelegt und der Zeichensatz (Kollation) wurde mit UTF-8 (um die griechischen Buchstaben anzeigen zu können) definiert.

Im Attribut „zahlenwert“ sollen alle berechneten Werte des jeweils dazugehörigen Namens gespeichert werden. Da es sich dabei um einzelne Zahlen handelt, wurde als Typ „Integer“ gewählt.

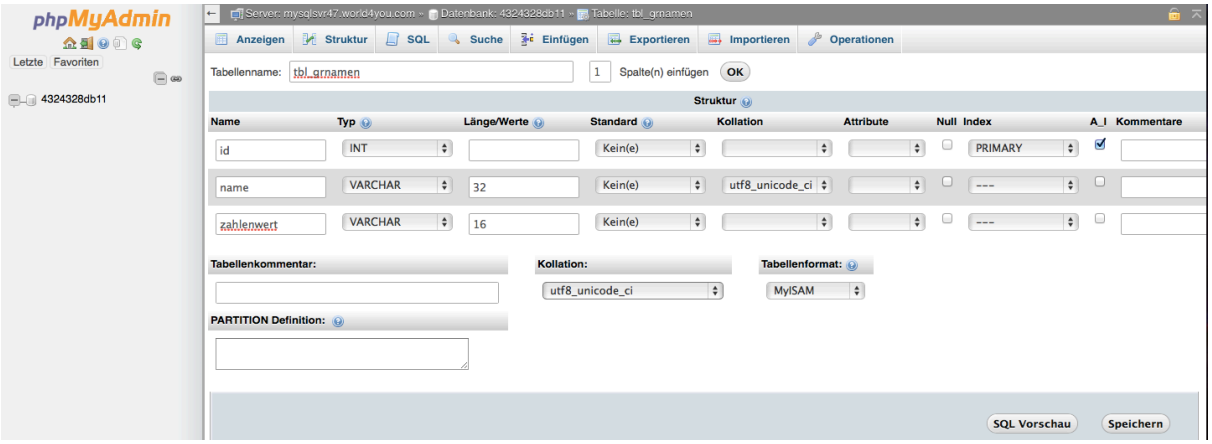


Abbildung 36 Attribute in tbl_grnamen

In MySQL ist es möglich, schon vorhandene csv-Dateien zu importieren. Dazu klickt man bei phpMyAdmin den Menüpunkt „Importieren“ an und wählt die Datei aus (siehe Abb. 37), sowie das Dateiformat, welches sie hat (siehe Abb. 38). Durch diesen Vorgang war es möglich, die Liste der griechischen Namen erfolgreich und unkompliziert in die Datenbank einzufügen, wie in Abb. 39 ersichtlich.

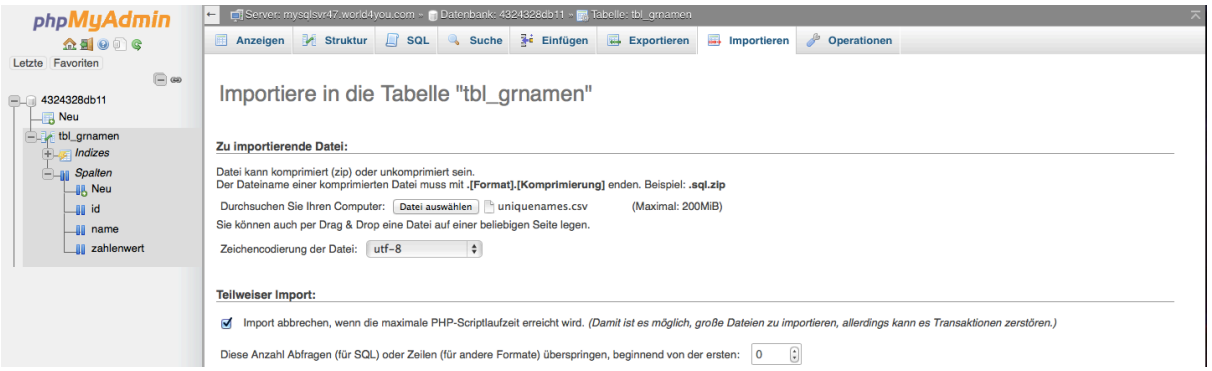


Abbildung 9 Importiervorgang csv-Datei (Teil 1)

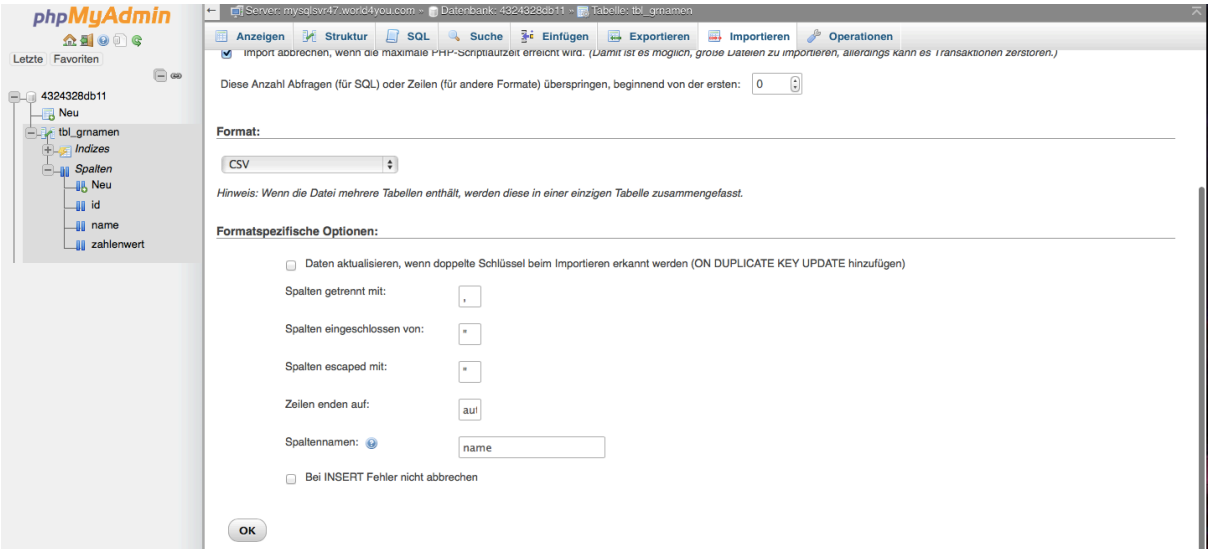


Abbildung 38 Importiervorgang csv-Datei (Teil 2)

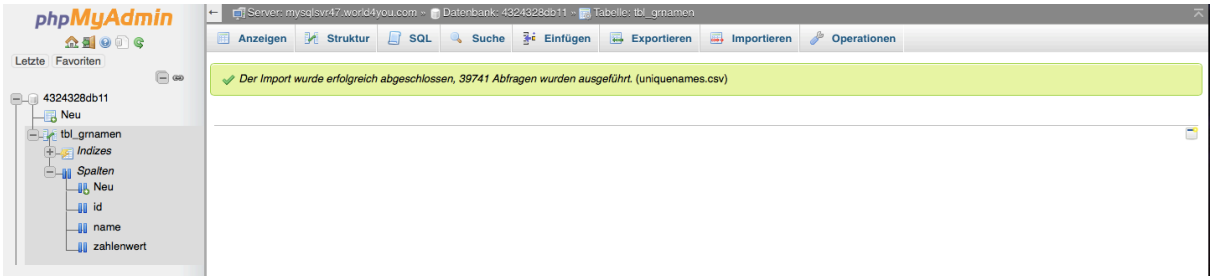


Abbildung 39 Importiervorgang csv-Datei (Teil 3)

Diese soeben in die Datenbank eingefügte Liste enthielt jedoch auch unbrauchbare Datensätze, welche mittels SQL-Befehl angezeigt (siehe Abb. 40) und im nächsten Schritt gelöscht wurden (siehe Abb. 41).

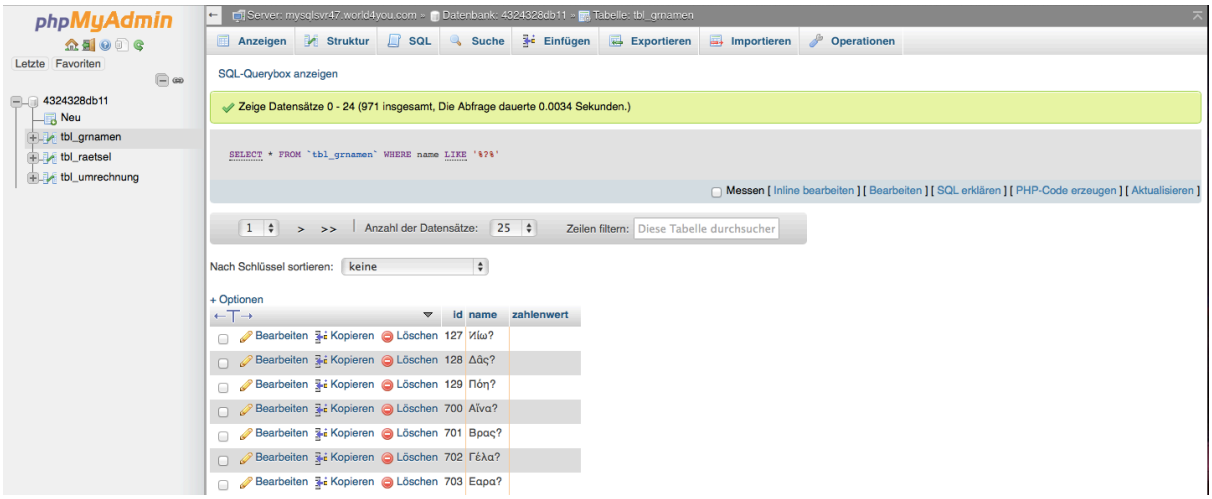


Abbildung 40 Abfrage auf Datensätze mit „?“ im Namen

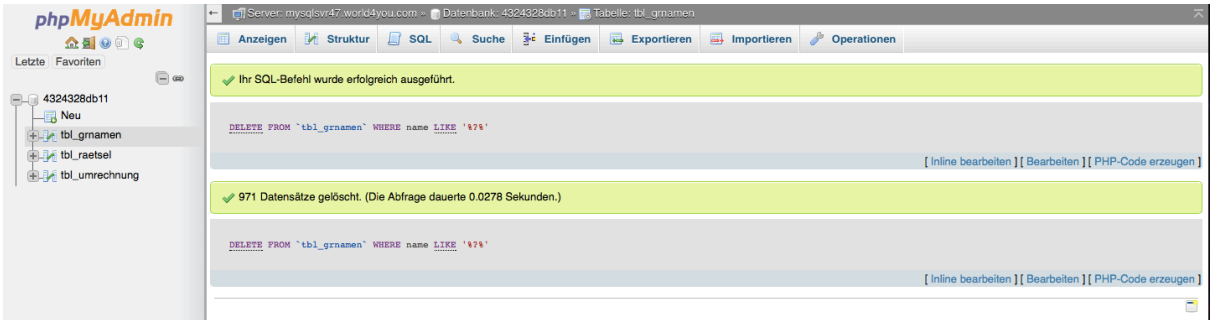


Abbildung 41 Alle Datensätze aus der Datenbank löschen, die ein „?“ enthalten

Doch auch nach der Löschung dieser unvollständigen Namen waren immer noch Datensätze vorhanden, von welchen kein Zahlenwert berechnet werden konnte (wegen eines oder mehrerer unbekannten Zeichen, siehe „4.5.2 Code Dokumentation“), wie an folgender Abbildung (Abb. 42) ersichtlich:

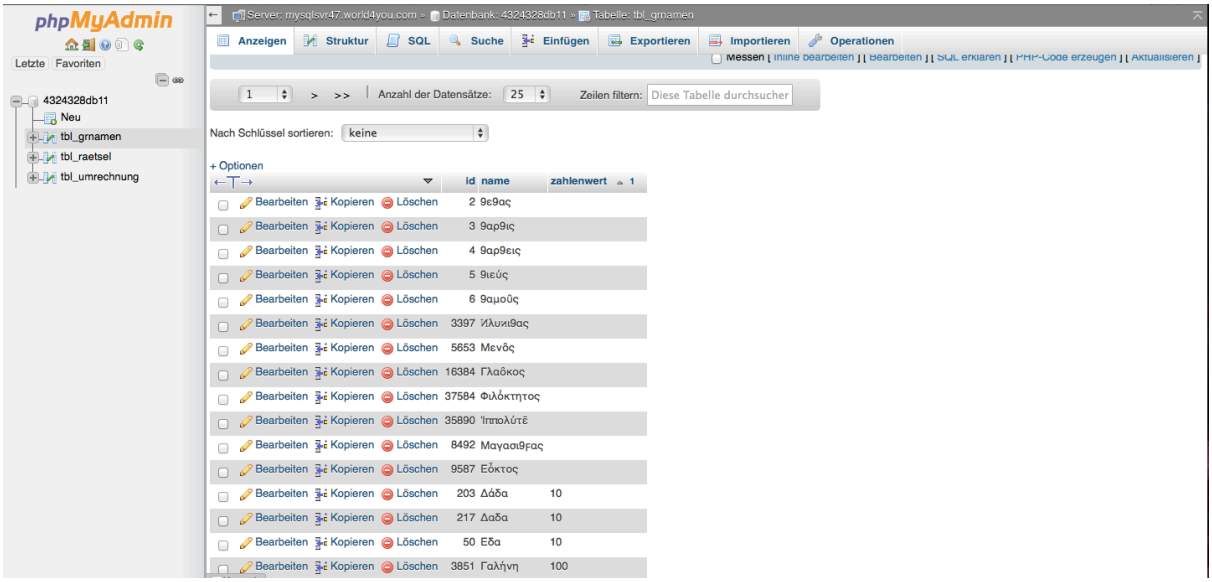


Abbildung 42 Problematische Datensätze

Das Problem dieser Datensätze war, dass sie Zeichen enthielten, die noch nicht in der Umrechnungstabelle gespeichert waren. Diese mussten dann händisch hinzugefügt werden. Abb. 43 zeigt dieselbe Abfrage noch einmal – nach der Hinzufügung. Hier kann man erkennen, dass nun zu jedem Namen ein Zahlenwert berechnet werden konnte.

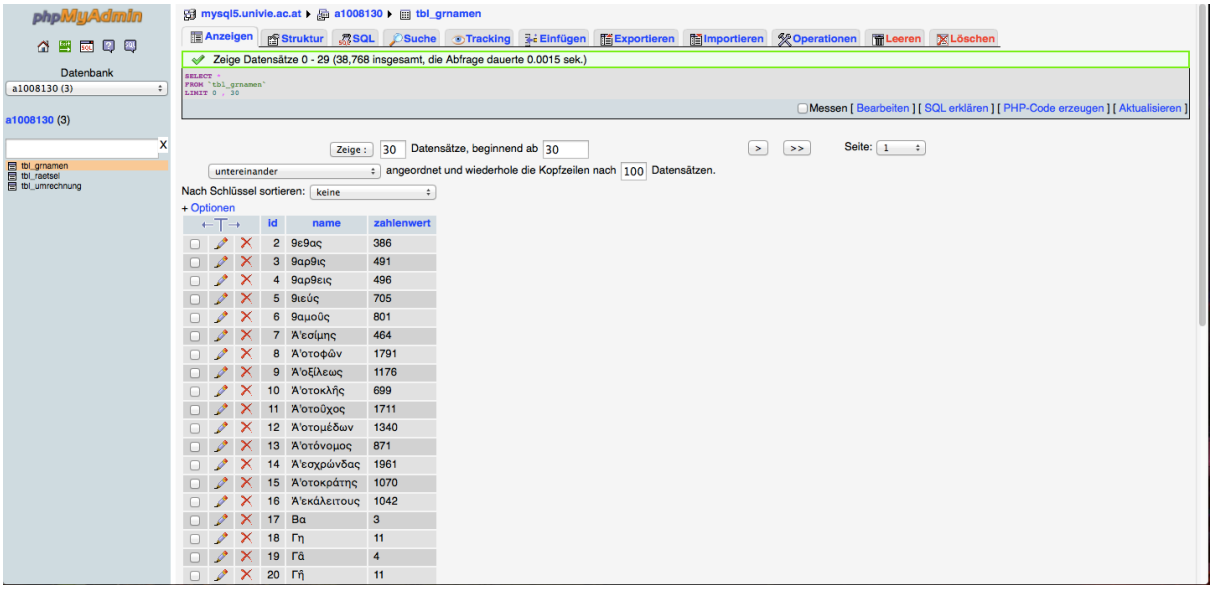


Abbildung 43 Jeder Datensatz ist nun vollständig (er hat eine ID, einen Namen und einen Zahlenwert) – alphabetisch geordnet

Tabelle der bekannten Rätsel: „tbl_raetsel“

Diese Tabelle enthält die folgenden fünf Attribute: „id“, „raetsel_gr“, „raetsel_de“, „zahlenwert“ und „ausnahme_sigma“. Weiters beinhaltet diese die vier bekannten Rätsel, die mir Herr ao. Univ.-Prof. Dr. Taeuber zur Verfügung gestellt hat und welche auf der Startseite des Web Interfaces angezeigt werden (jeweils mit der berechneten Lösung, wie bereits angesprochen) – Abb. 44 zeigt alle gespeicherten Rätsel in der Tabelle „tbl_raetsel“.

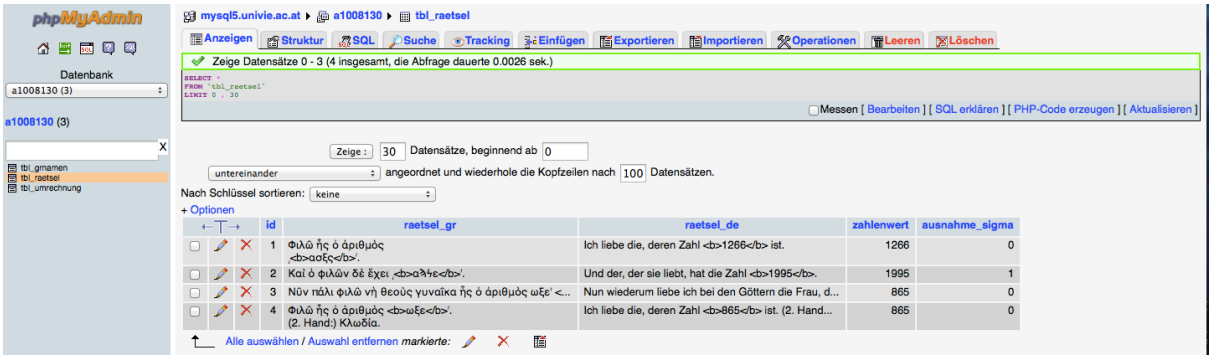


Abbildung 44 Tabelle mit den bekannten Rätseln („tbl_raetsel“)

Der Primärschlüssel dieser Tabelle ist ebenfalls das Attribut „id“. Wie auch schon in der vorigen Tabelle wurde es mit der Eigenschaft „AUTO_INCREMENT“ erstellt. In den Datenbankattributen „raetsel_gr“ und „raetsel_de“, beide als Typ „Text“ definiert, werden jeweils das anzuzeigende deutsche bzw. griechische Rätsel gespeichert. Weiters, wie in der Tabelle tbl_grnamen schon erwähnt, wurde ein Attribut „zahlenwert“ erstellt, in welches die Zahl des zu berechnenden Rätsels gespeichert wird. Als Letztes gibt es noch das Attribut „ausnahme_sigma“ – hierbei handelt es sich um den Typ „Boolean“ (1 und 0 in der Datenbank). Dieser Wert wird bei der Berechnung des Zahlenwertes verwendet, denn so wird, wenn das Attribut den Wert 1 hat, die Ausnahme des vertauschten Sigmas (siehe „4.3.2 Besondere Anforderungen“) in die Berechnung miteinbezogen. Dies ist bei einem der bekannten Rätsel von Bedeutung. Abb. 45 zeigt die Tabelle „tbl_raetsel“:

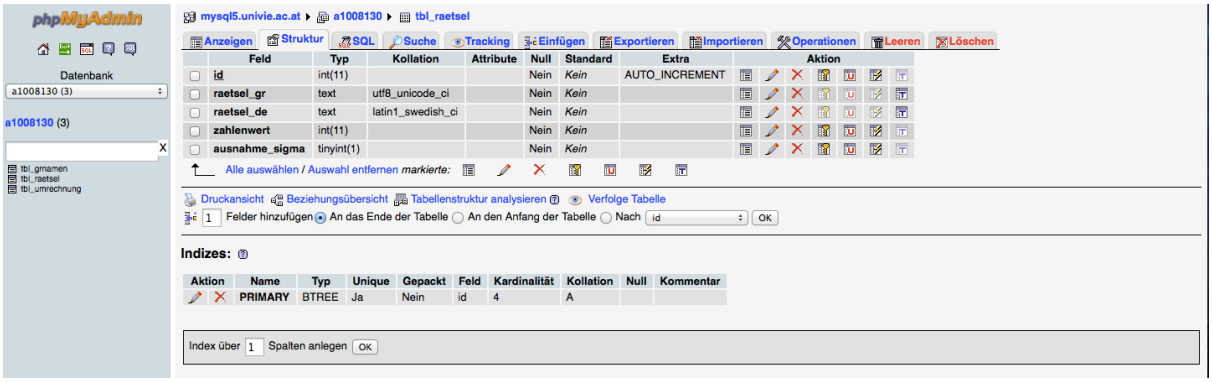


Abbildung 45 Attribute von „tbl_raetsel“

Umrechnungstabelle: „tbl_umrechnung

Diese Tabelle besitzt folgende drei Attribute: „id“, „buchstabe“ und „zahlenwert“. Es wurden darin alle griechischen Buchstaben (Groß- und Kleinbuchstaben und alle möglichen Varianten der Vokale, also inkl. Akzente und Spiritus) mit dem jeweiligen Zahlenwert eingefügt. Akzente vor Großbuchstaben (da auf ihnen kein Platz dafür ist) wurden ebenfalls eingetragen (und damit Fehler abgefangen, siehe „4.5.2 Code Dokumentation“). Diesen wurde der Zahlenwert 0 zugeordnet. Problematisch gestaltete es sich beispielsweise mit dem

„Buchstaben“ „9“ – nach Absprache mit Herrn Univ.-Prof. Taeuber handelte es sich dabei wohl um ein Koppa und deshalb sollte diesem der Wert 90 zugeordnet werden. Abb. 46 zeigt einen Teil der Umrechnungstabelle „tbl_umrechnung“.

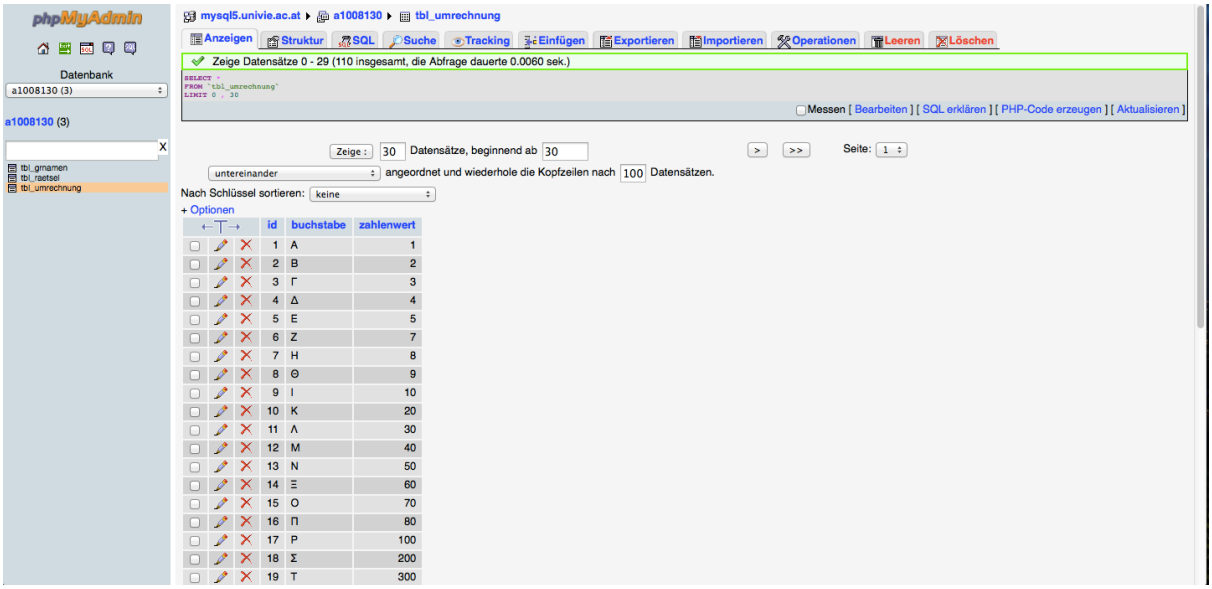


Abbildung 46 Umrechnungstabelle („tbl_umrechnung“)

Primärschlüssel dieser Tabelle ist wieder das Attribut „id“, welches auch wieder mit der Eigenschaft „AUTO_INCREMENT“ erstellt wurde. Im Attribut „buchstabe“ werden die einzelnen griechischen Buchstaben gespeichert. Es wurde der Typ „Varchar“ mit der Länge 2 gewählt, um wirklich alle griechischen Buchstaben (auch mit Spiritus) abbilden zu können. In dieser Tabelle wird das Attribut „zahlenwert“ mit dem Typ „Integer“ dafür verwendet, um jedem Buchstaben eine eindeutige Zahl für die Umrechnung zuzuordnen. Abb. 47 zeigt die Attribute der Tabelle „tbl_umrechnung“:

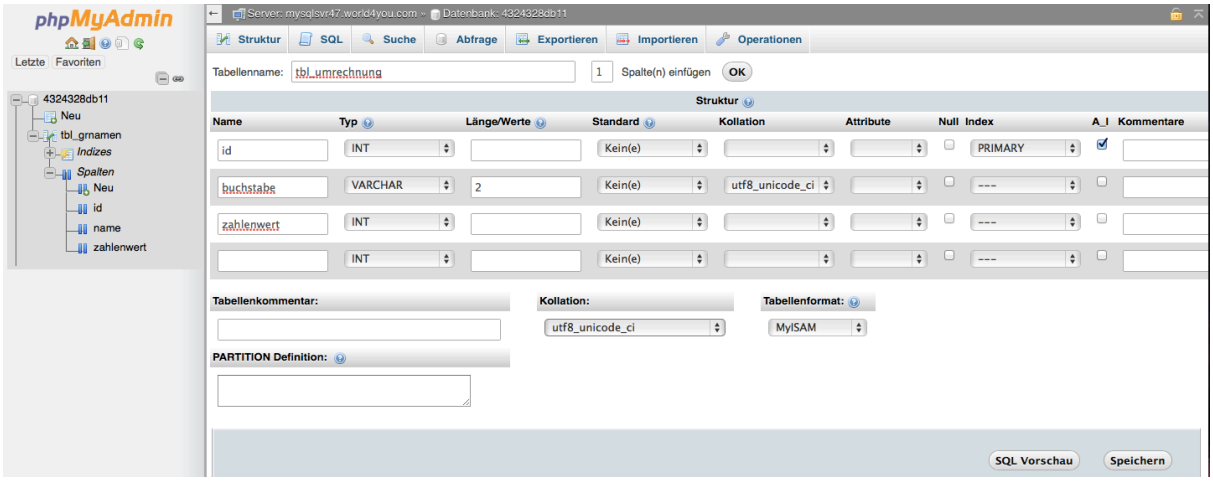


Abbildung 47 Attribute „tbl_umrechnung“

In den folgenden Abbildungen (Abb. 48-50) sieht man, wie die Tabelle „tbl_umrechnung“ mit Datensätzen aus der Umrechnungstabelle befüllt wird.

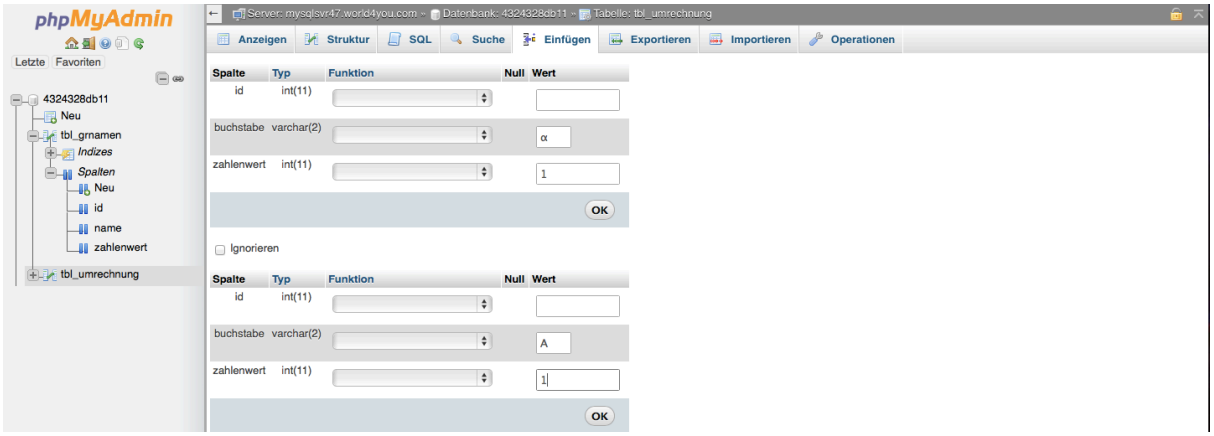


Abbildung 48 Einfügen der Buchstaben und Zahlenwerte der Umrechnungstabelle in „tbl_umrechnung“ (Teil 1)

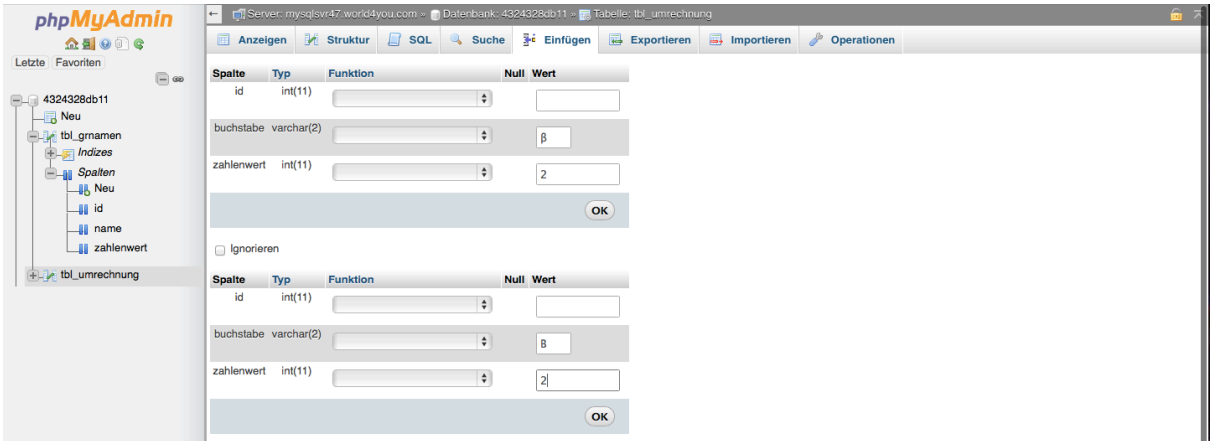


Abbildung 49 Einfügen der Buchstaben und Zahlenwerte der Umrechnungstabelle in „tbl_umrechnung“ (Teil 2)

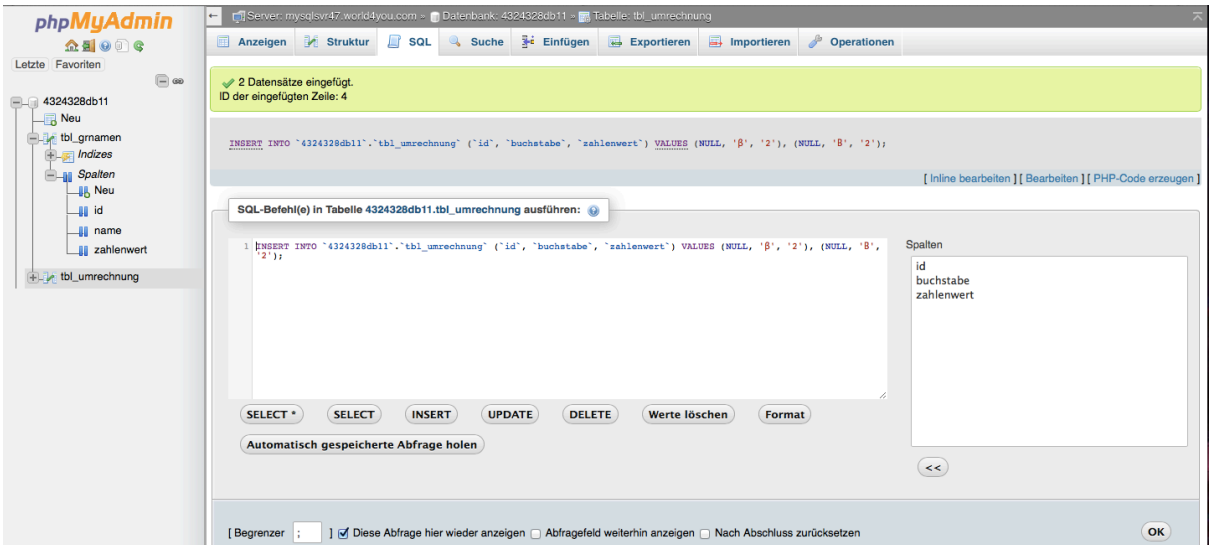


Abbildung 50 Einfügen der Buchstaben und Zahlenwerte der Umrechnungstabelle in „tbl_umrechnung“ (Teil 3)

In Abbildung 51 sieht man alle Ausprägungen des kleinen Alphas, nachdem sie händisch eingegeben wurden. Dieses Prozedere musste bei allen Vokalen durchgeführt werden und auch bei Ausnahmen, wie z.B. dem „Buchstaben“ „9“ (altes Zeichen für Koppa, wie bereits angesprochen), welcher den Wert 90 erhielt (siehe Abb. 52).

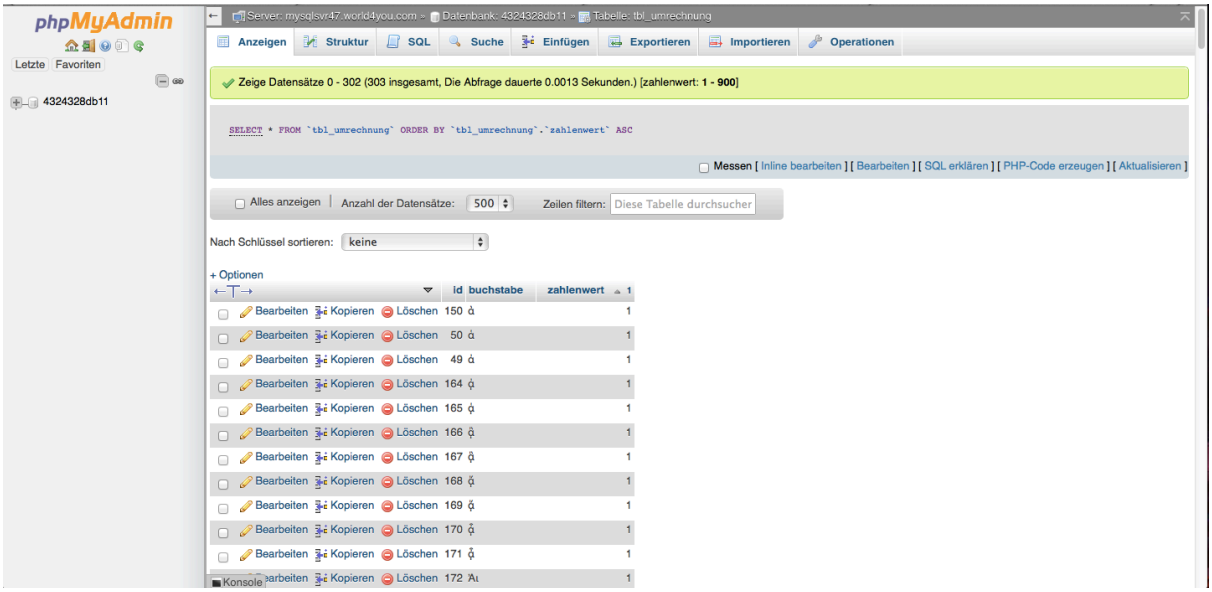


Abbildung 51 Umrechnungstabelle „tbl_umrechnung“ mit allen Ausprägungen des kleinen Alphas

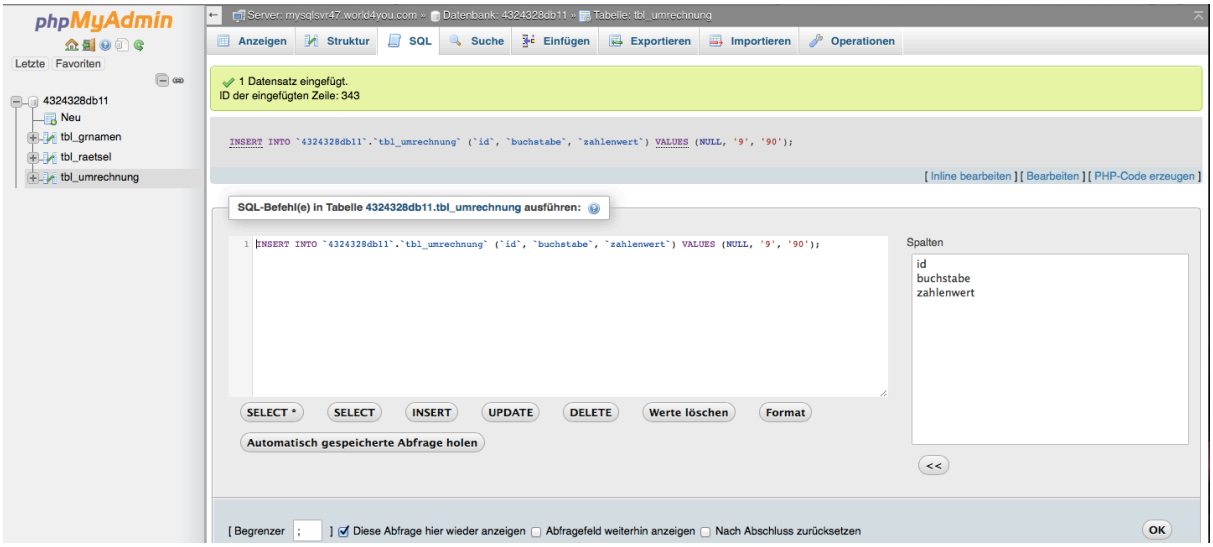


Abbildung 52 Buchstabe „9“ (Koppa) Wert 90 zuordnen

Hinweis: Ein seit kurzem verfügbares Toolset für interaktive Webseiten für Datenbankabfragen und Auswertungen ist der, auf der Programmiersprache R beruhende, Shiny-Server. Genaueres dazu findet man auf: http://shiny.rstudio.com.

4.5 Web Interface

4.5.1 Implementierung

Provisorische Startseite und erste Tests

Zuerst wurde eine provisorische Startseite des Web Interfaces mittels HTML erstellt und mittels PHP versucht, eine Datenbankverbindung aufzubauen. Da dies auf Anhieb funktioniert hat, konnten auch erste Tests mit SQL-Statements durchgeführt werden. Allerdings ergaben sich bezüglich der Darstellung Probleme – so wurden nicht bekannte Zeichen (in diesem Fall griechische Buchstaben) mit einem Fragezeichen dargestellt. Deshalb wurde die UTF-8 Kodierung in den Code implementiert, sodass es danach zu keinerlei Darstellungsprobleme mehr gekommen ist, wie in den beiden Abb. 53 und 54 ersichtlich.

Abbildung 53 Provisorische Startseite des Web Interfaces (Teil 1)

Abbildung 54 Provisorische Startseite des Web Interfaces (Teil 2)

CSS und Menüstruktur

Als Nächstes wurde eine Cascading Style Sheet-Datei (CSS-Datei) und ein Menü für das Web Interface erstellt und eingebunden. Damit sieht das Web Interface gleich einladender aus (siehe Abb. 55). Neben der Startseite lassen sich nun auch noch drei weitere Seiten mit folgenden Namen finden: „Über dieses Projekt“, „Über die Rätsel“ und „Impressum“.

Abbildung 55 Web Interface mit eingebundenem CSS und Menü

Anzeige der Ergebnisse

Der nächste Schritt beinhaltete div-Tags rund um die Ergebnisse der SQL-Abfrage – erkennbar an einer Box mit eigenem Scrollbalken (siehe Abb. 56). Diese Lösung wurde zum Zweck der Übersichtlichkeit gewählt.

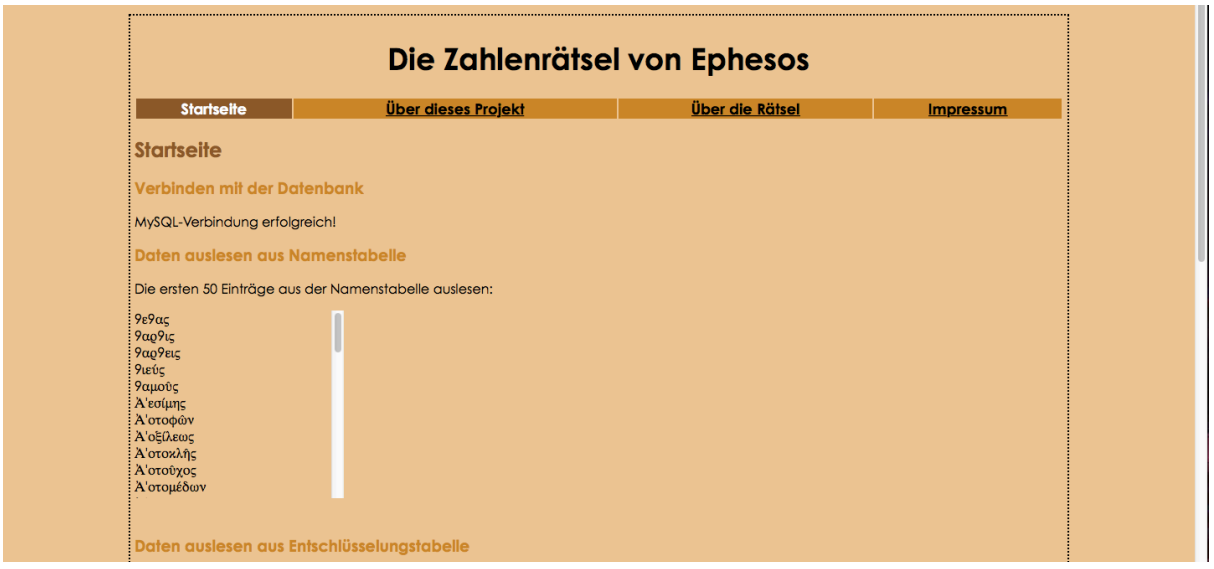


Abbildung 56 div-Tags zum Gruppieren der Ergebnisse

Drei neue Seiten

Im weiteren Verlauf wurden die restlichen drei Seiten („Über dieses Projekt“, „Über die Rätsel“ und „Impressum“) vorerst provisorisch, mit der Zeit dann mit passenden Inhalten befüllt (siehe Abb. 57-59).

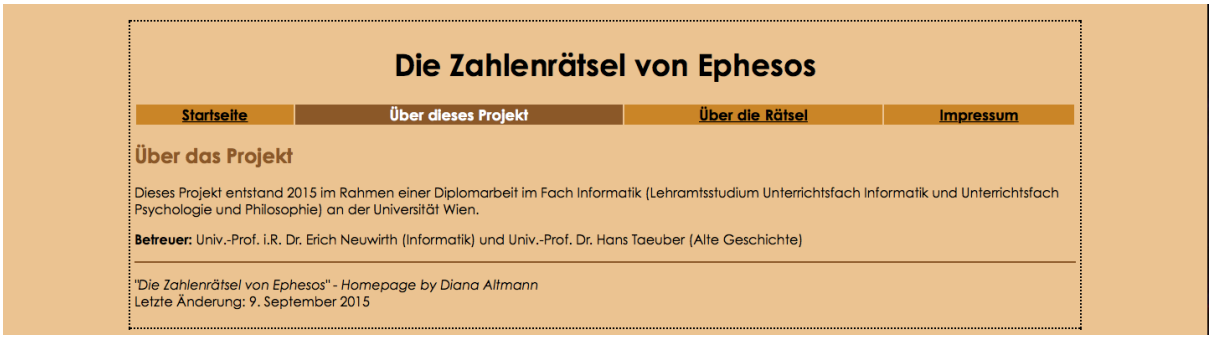


Abbildung 57 Seite „Über dieses Projekt“ mit Inhalt

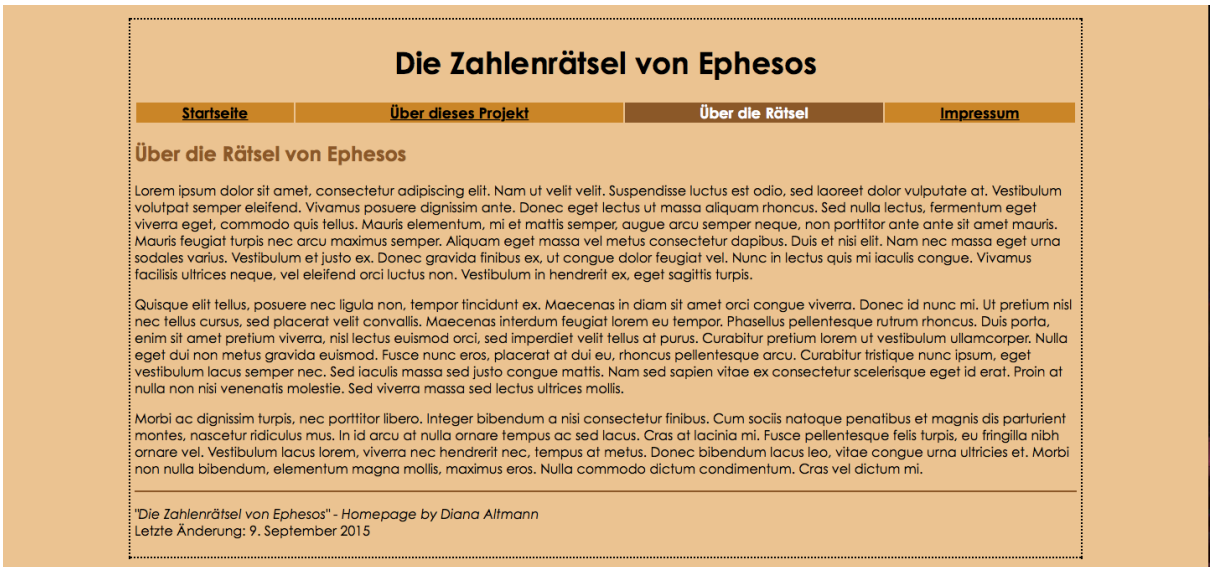


Abbildung 58 Seite „Über die Rätsel“ mit provisorischem Inhalt

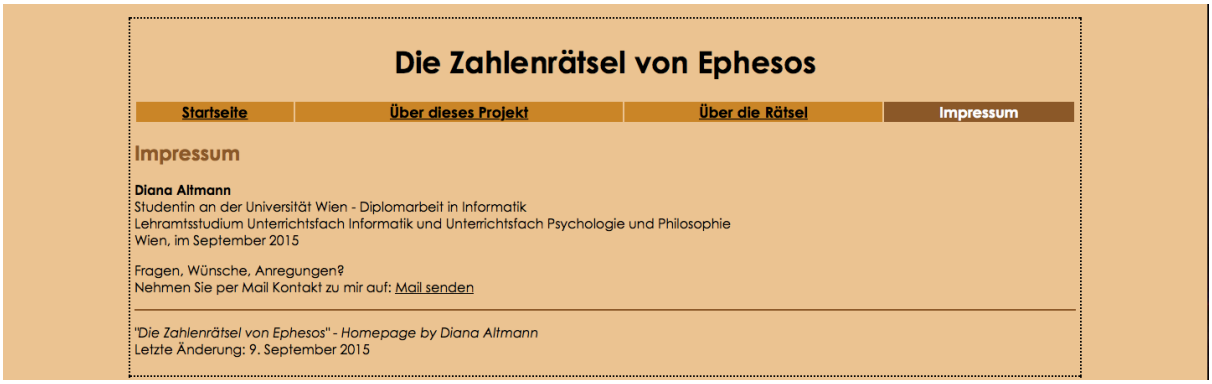


Abbildung 59 Seite „Impressum“ mit Inhalt

Adminbereich

Als Nächstes wurde ein Adminbereich erstellt, in welchem diverse Tests und Skripts durchgeführt wurden (wie z.B. das Skript zum Berechnen und Speichern der Zahlenwerte in

die Datenbank). Man gelangt mittels Link (ganz unten) auf der Startseite des Web Interfaces zum Adminbereich (siehe Abb. 60), welcher kennwortgeschützt ist (es wird eine Session angelegt, Erklärung siehe „4.5.2 Code Dokumentation“). Das Login zum Adminbereich ist in Abb. 61 ersichtlich.

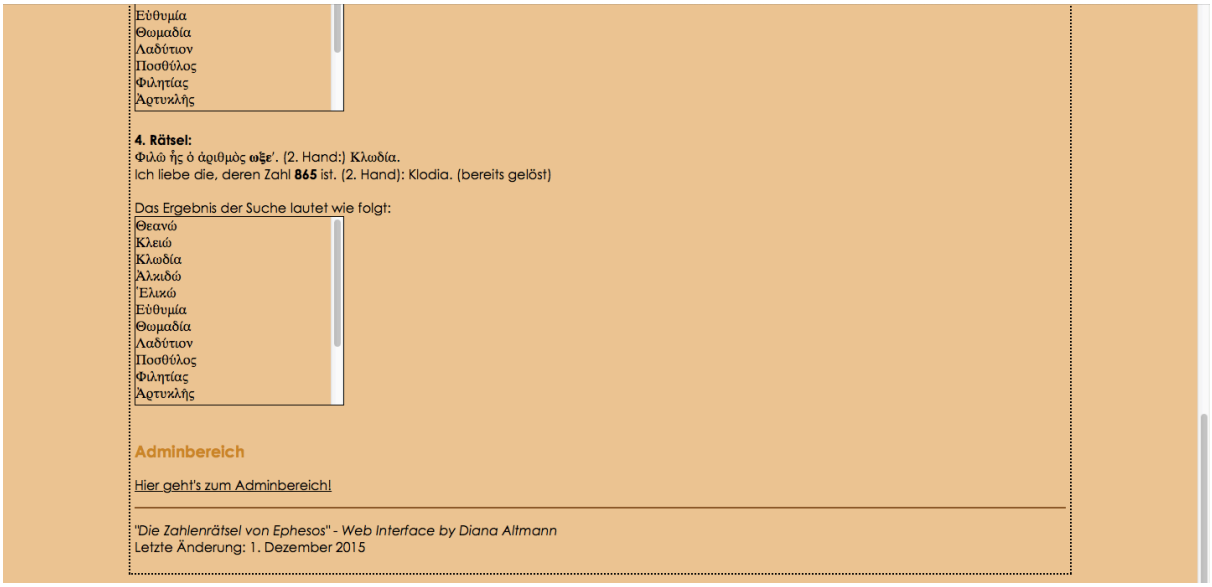


Abbildung 60 Link zum Adminbereich

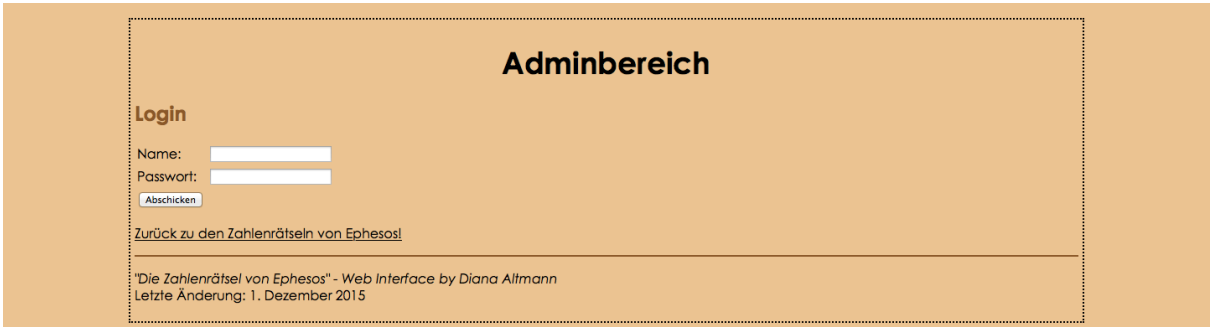


Abbildung 61 Login zum Adminbereich

Der Adminbereich besteht aus drei Seiten, welche oben im Menü sichtbar sind. Die ersten beiden Seiten (siehe Abb. 62 und 63) sind dazu da, um die Verbindung zur Datenbank zu testen und Daten auszulesen. Weiters werden einige Tests mittels Datenabfragen durchgeführt.

Abbildung 62 Startseite Adminbereich („Zahlenwerte berechnen“)

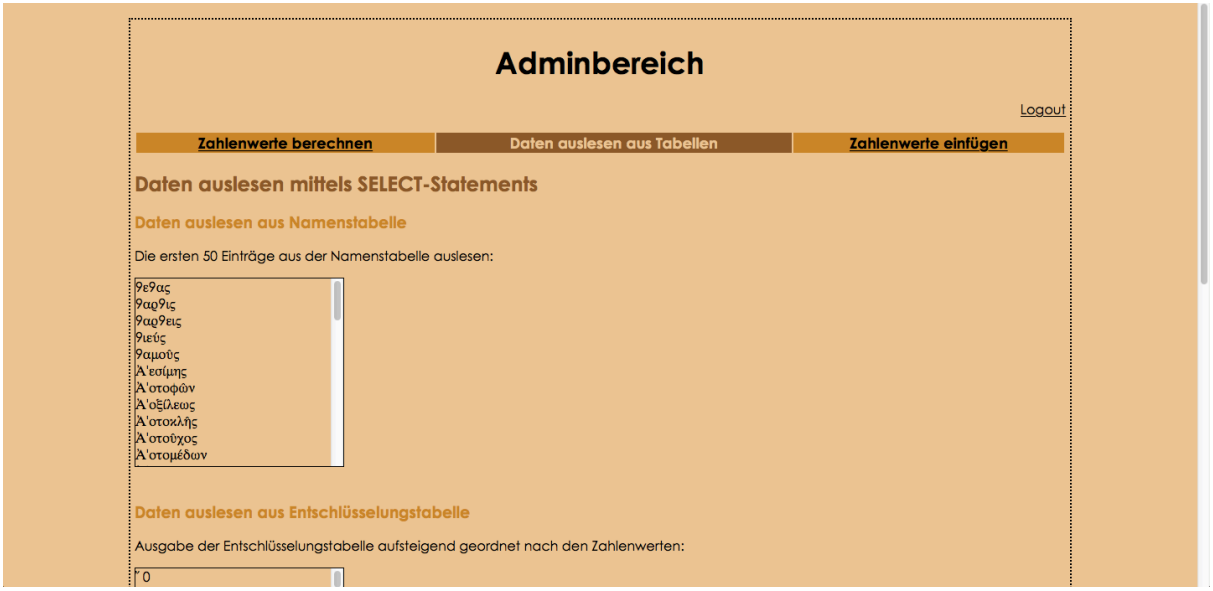


Abbildung 63 Seite 2 des Adminbereichs („Daten auslesen aus Tabellen“)

Nachdem man den Logout-Link gedrückt hat, wird man automatisch zur Login-Maske des Adminbereichs weitergeleitet (siehe Abb. 64). Mittels Klick auf „Zurück zu den Zahlenrätsel von Ephesos“ gelangt man wieder auf die Startseite des Web Interfaces.

Abbildung 64 Nach dem Klicken auf den Logout-Button gelangt man wieder zur Login-Seite

Implementierung von „Zahl eingeben“

Als Nächstes wurde ein Feld namens „Zahl“ auf der Startseite implementiert, in welches man eine beliebige Zahl eintragen und abschicken kann und man daraufhin vom Programm alle Namen als Ergebnisse ausgegeben bekommt, die denselben Zahlenwert haben (welcher zuerst mit Hilfe eines Skripts berechnet und in die Datenbank zu dem jeweiligen Datensatz gespeichert wurde – Erklärung siehe „4.5.2 Code Dokumentation“), wie die eingegebene. Veranschaulicht wird dieser Vorgang in Abb. 65.

Abbildung 65 Ergebnisse nach Abschicken der Zahl 1000

Bekannte Rätsel werden auf der Startseite eingebunden

Neben dieser neuen Funktionalität wurden nun auch die vier bekannten Rätsel aus der Tabelle „tbl_raetsel“ auf der Startseite eingebunden (jeweils auf Griechisch und Deutsch) und gelöst, wie dies an Abb. 66 und Abb. 67 zu sehen ist.

Abbildung 66 Bekannte Rätsel werden in die Startseite eingebunden (Teil 1)

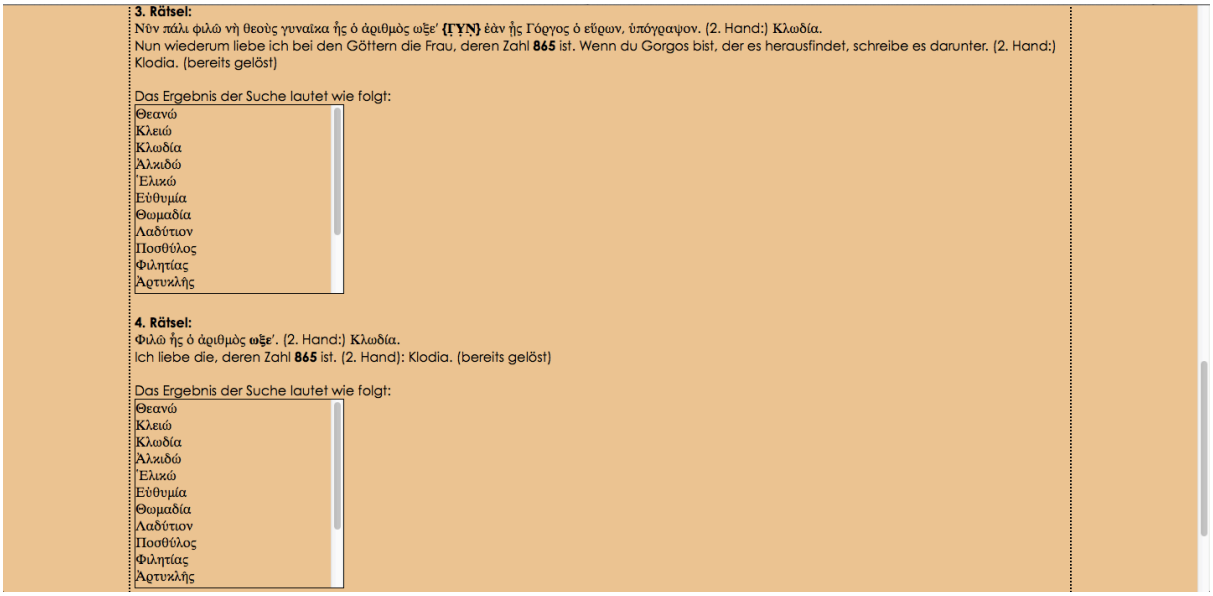


Abbildung 67 Bekannte Rätsel werden in die Startseite eingebunden (Teil 2)

Implementierung von „Name eingeben“

Im weiteren Verlauf wurde ein Feld namens „Name“ auf der Startseite implementiert, in welches man einen beliebigen griechischen Namen eingeben und abschicken kann und man daraufhin vom Programm den Zahlenwert ausgegeben bekommt, inklusive der Aufschlüsselung der Berechnung, siehe Abb. 68 und Abb. 69.

Abbildung 68 Feld „Name“ wurde hinzugefügt

Abbildung 69 Feld „Name“ wurde abgeschickt – Zahlenwert und Zahlenwertberechnung sind ersichtlich

Klickbarer Keyboardersatz

Einem hilfreichen Tipp von Herrn ao. Univ.-Prof. Dr. Neuwirth nachgehend, wurde als nächster Schritt ein klickbarer Keyboardersatz implementiert, welcher beim Ausfüllen des Feldes „Name“ helfen soll, diesen mit griechischen Buchstaben einzugeben ohne griechische Tastatur (denn ohne passende Tastatur war dies bis dato nicht möglich außer mittels Copy-Paste). Dazu wurde eine Javascript-Datei eingebunden, die ausgeführt wie folgt (Abb. 70 und Abb. 71) aussieht:

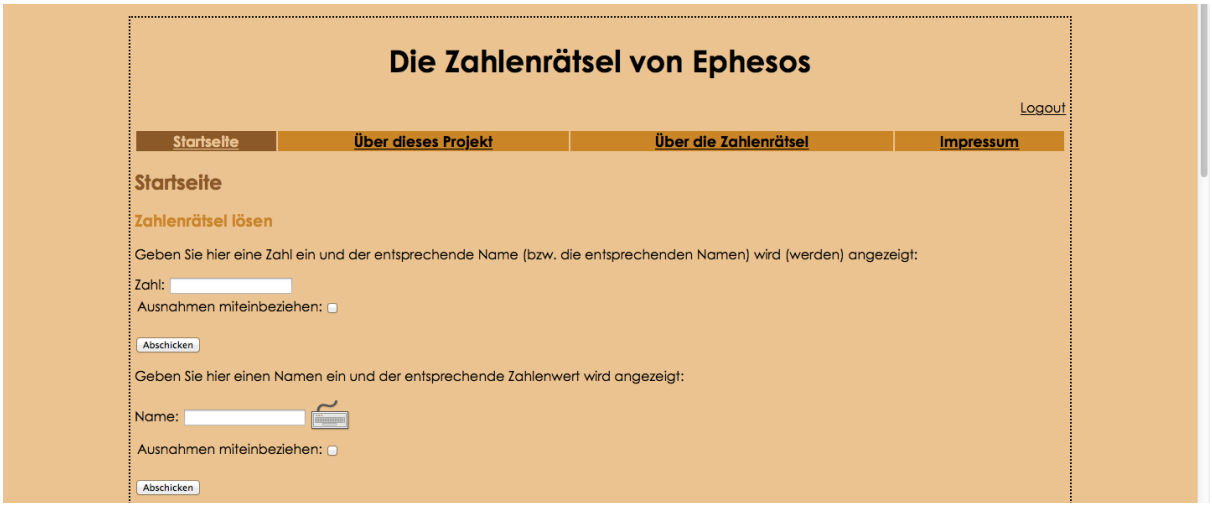


Abbildung 70 Klickbarer Keyboardersatz

Abbildung 71 Klickbarer Keyboardersatz in Action

Implementierung der Ausnahmen

Wie bereits im Kapitel „4.3 Umrechnungstabelle“ angesprochen, war es auch Teil des Projekts, verschiedene Ausnahmen miteinzubeziehen (ς = σ, αι = ε, ει = ι, ι = ει, η = ι, ω = ο, ιος = ις, ιον = ιν, ου [vor Vokal] = β, β [vor Vokal] = ου). Da es möglich sein sollte, diese

beliebig miteinander zu kombinieren, stellte dies jedoch einen erheblichen Rechenaufwand dar und so kam es in der ersten Phase zu Timeouts beim Server.

Mit Hilfe von Arrays konnte eine erhebliche Verbesserung der Laufzeit erzielt und damit die Timeouts vermieden werden. Nach dieser Optimierung (siehe „4.5.2 Code Dokumentation“) funktionierte dann alles wie gewünscht und so war es nun möglich, sowohl bei „Zahl“ als auch bei „Name“ Ausnahmen (in jeder beliebigen Kombination) miteinzubeziehen. Wird das Häkchen bei „Ausnahmen miteinbeziehen“ nicht gesetzt, so werden die Ergebnisse wie gewöhnlich berechnet (wie in der Umrechnungstabelle festgelegt) und angezeigt. In den folgenden Abbildungen (Abb. 72-75) ist eine Eingabe mit Ausnahmen – zuerst bei „Zahl“, dann bei „Name“ – zu sehen.

Abbildung 72 Eingabe der Zahl „100“ mit Ausnahmen (Teil 1)

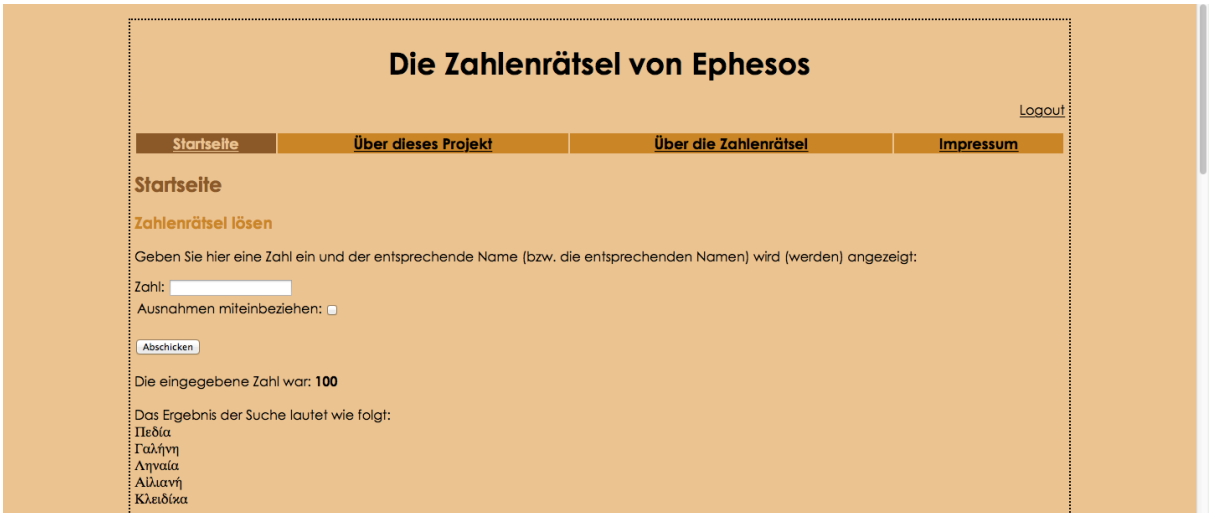


Abbildung 73 Eingabe der Zahl „100“ mit Ausnahmen (Teil 2)

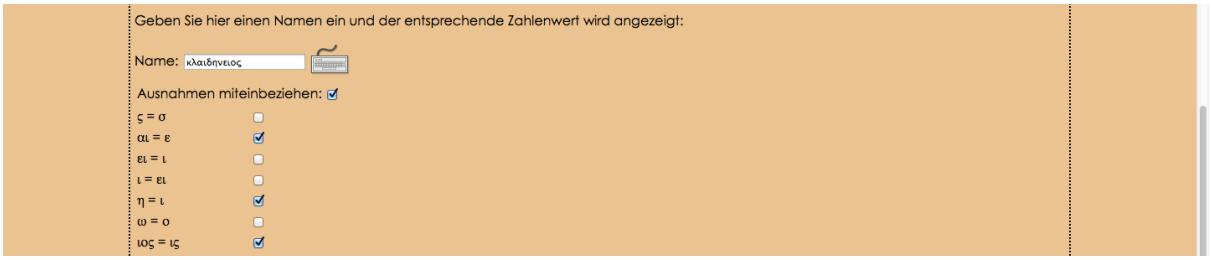


Abbildung 74 Eingabe eines Namens mit Ausnahmen (Teil 1)

Abbildung 75 Eingabe eines Namens mit Ausnahmen (Teil 2)

Kennwortschutz für das gesamte Web Interface

Einer Bitte von Herrn ao. Univ.-Prof. Dr. Taeuber nachkommend, wurde im nächsten Schritt das gesamte Web Interface kennwortgeschützt (erneut mittels einer Session-Variablen analog zum Passwortschutz des Adminbereichs; bei Abschluss dieser Arbeit jedoch nicht mehr relevant). Ruft man die Seite also wie gewohnt über die URL http://homepage.univie.ac.at/a1008130 auf, so wird folgender Login-Bildschirm angezeigt anstatt der Startseite (Abb. 76):

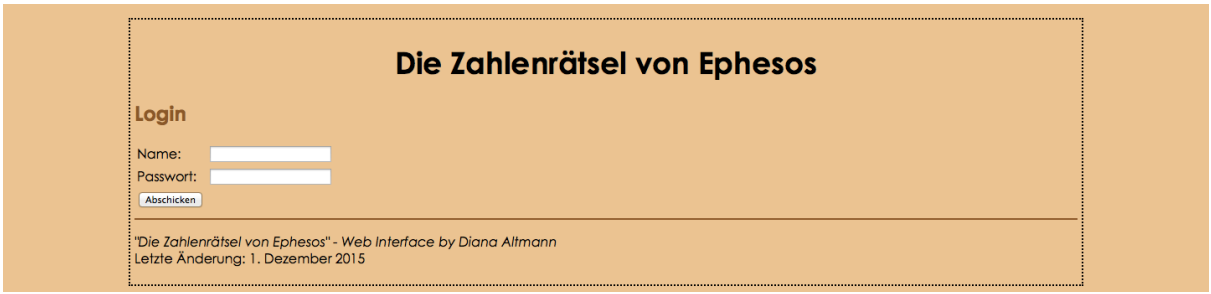


Abbildung 76 Login-Bildschirm

Nach Eingabe der richtigen Kombination aus Name und Passwort gelangt man dann zur bereits bekannten Startseite des Web Interfaces (siehe Abb. 77).

Abbildung 77 Startseite des Web Interfaces

Klickt man rechts oben (analog zum Logout-Link im Adminbereich) auf „Logout“, so gelangt man erneut zur Login-Maske (wie in Abb. 76 bereits gezeigt).

Text über die Zahlenrätsel von Ephesos

Ein weiteres Update war das Einbinden eines Textes über die Zahlenrätsel in Ephesos, welchen mir dankenswerterweise Herr ao. Univ.-Prof. Dr. Taeuber zur Verfügung gestellt hat (siehe Abb. 78):

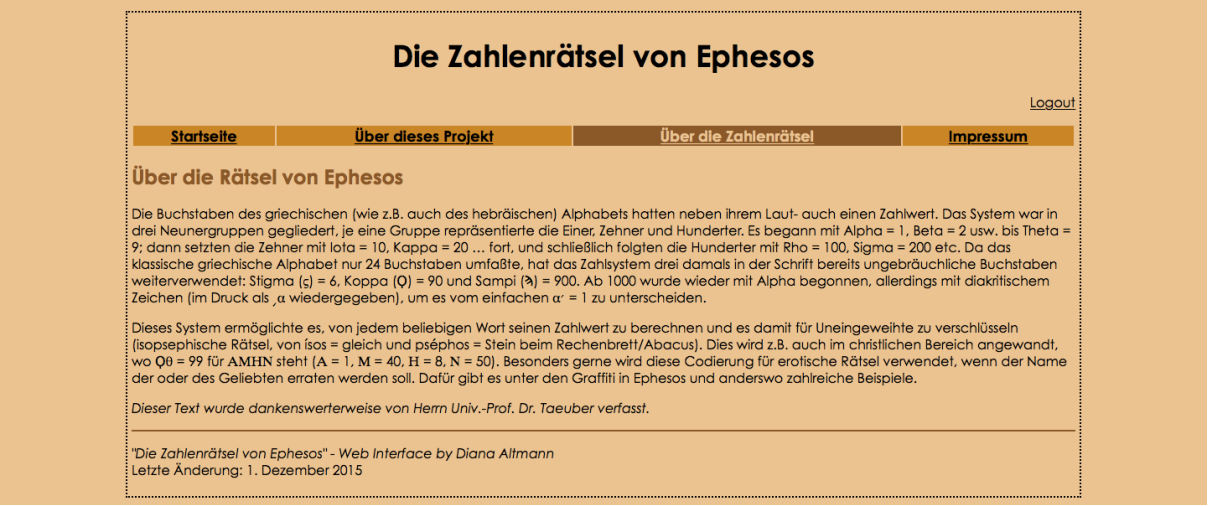


Abbildung 78 Inhalt hinzugefügt (Seite „Über die Zahlenrätsel“)

4.5.2 Code Dokumentation

In diesem Unterkapitel werden alle für das Projekt benötigten bzw. bei der Programmierung erstellten Dateien beschrieben. Es wird dabei auch auf die Evolution des Codes eingegangen (z.B. in Bezug auf Optimierungen).

Folgende Dateien wurden für das Projekt angelegt und befinden sich auf beiliegender CD-ROM:

- loginindex.php
- db.php
- index.php
- menue.php
- ueberprojekt.php
- ueberraetsel.php
- impressum.php
- footer.php
- styles.css
- login.php
- admin.php
- admin2.php
- insert.php
- adminmenue.php
- keyboard2.js
- keyboard2.js.css

Datei „loginindex.php“

Die „loginindex.php“-Seite ist dazu da, um der Startseite (und allen weiteren Seiten) des „Zahlenrätsel von Ephesos“-Web Interfaces ein zwingendes Login vorzuschalten (wurde am Ende der Diplomarbeit jedoch nicht mehr benötigt).

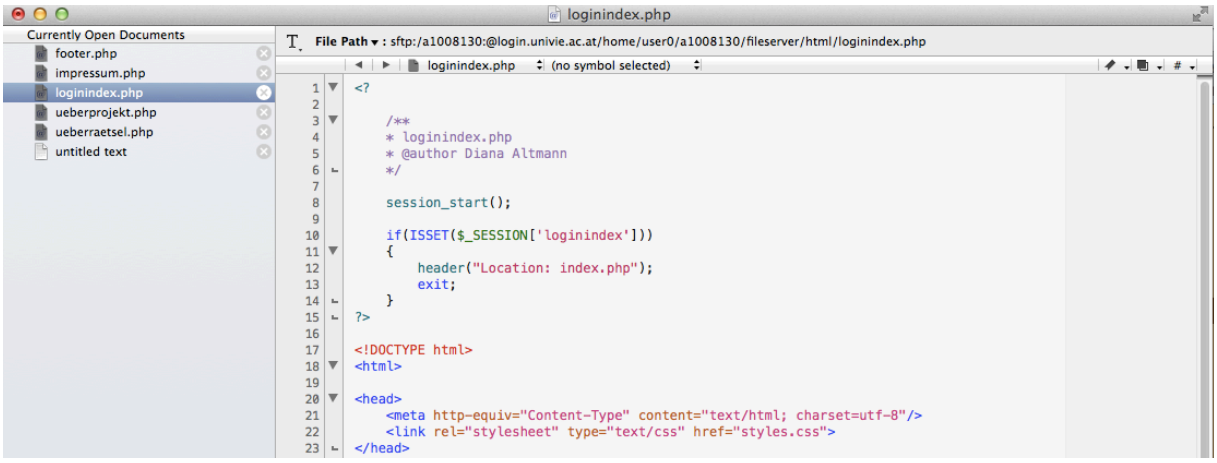


Abbildung 79 Code „loginindex.php“ (Teil 1)

Der PHP-Code beginnt mit einem Kommentar (Inhalt sind Name der Datei und Name der Autorin). Danach wird eine Session erzeugt und im `if`-Block überprüft, ob die Session bereits gesetzt ist – falls ja, wird automatisch zur Startseite des Web Interfaces weitergeleitet („index.php“).

Als Nächstes wird der Doctype mittels `<!DOCTYPE html>` definiert und die Grundstruktur einer HTML-Seite eingebettet (dementsprechend, wie an Abb. 79 und Abb. 80 ersichtlich, `<html>`, `<head>`, `</head>`, `<body>`, `</body>`, `</html>`). Im `<head>`-Bereich wird mittels Meta-Tag das Charset (die Zeichencodierung) auf UTF-8 festlegt. Die darauffolgende Codezeile bindet die CSS-Datei namens „styles.css“ ein.

Abbildung 80 Code „loginindex.php“ (Teil 2)

Der `<body>`-Bereich beinhaltet zwei Überschriften: Die erste („Die Zahlenrätsel von Ephesos“) ist eingeschlossen in `<h1>`- und `</h1>`-Tags und analog dazu die zweite („Login“) in `<h2>`- und `</h2>`-Tags.

Danach wird ein Formular definiert, welches mit der Methode „post“ (Erklärung siehe „3.1.8 Vordefinierte Informationen“) abgeschickt wird und dessen Name „login“ ist. Es enthält zwei Felder („name“ und „password“), die dazu gedacht sind, den Usernamen und das dazugehörige Passwort einzugeben. Abgeschlossen wird das Formular mit dem Submit-Button, welcher es, wie im Formularkopf definiert, an die Seite „loginindex.php“ schickt.

Im ersten `if`-Block wird überprüft, ob das Formular abgeschickt wurde – wenn ja, wird auf Username und Passwort überprüft. Stimmen diese überein mit den Werten, die im Code eingetragen sind, wird, wenn noch keine Session gesetzt ist, diese mit `$_SESSION` initialisiert und der Wert „user“ wird zugewiesen. Mittels „Location: loginindex.php“ in der „header“-Methode wird die Seite noch einmal aufgerufen – dieses Mal jedoch mit gesetzter „user“-Session und es erfolgt automatisch die Weiterleitung auf „index.php“. Ist die Kombination aus Username und Passwort jedoch falsch, so wird eine Fehlermeldung („Zugangsdaten ungültig.“) ausgegeben.

Bevor der `<body>`-Bereich geschlossen wird, wird noch die Fußzeile („footer.php“) inkludiert und ausgegeben.

Datei „db.php“

In dieser Datei sind alle wichtigen Daten (Servername, Username und Passwort – jeweils gespeichert in den Variablen `$server`, `$user`, `$pass`) festgelegt, um sich mit der Datenbank zu verbinden.

Abbildung 81 Code „db.php“

Mittels „mysql_connect“ und der soeben genannten Daten wird dann eine Verbindung zur Datenbank hergestellt. Mit der Funktion „mysql_set_charset“ wird der Verbindungszeichensatz auf UTF-8 gesetzt, um eventuelle Darstellungsprobleme mit griechischen Buchstaben zu vermeiden. Als Nächstes wird die Datenbank, die benötigt wird, ausgewählt – falls diese nicht existiert, wird eine Fehlermeldung ausgegeben („Die Datenbank existiert nicht.“). Schlägt allgemein die Verbindung zur Datenbank fehl, so wird am Ende ebenfalls eine Fehlermeldung ausgegeben („MySQL-Verbindung leider nicht erfolgreich!“).

Datei „index.php“

In dieser Datei wird zuerst der Doctype mittels `<!DOCTYPE html>` definiert und die Grundstruktur einer HTML-Seite eingebettet. Im `<head>`-Bereich wird mittels des Meta-Tags das Charset (die Zeichencodierung) auf UTF-8 festlegt. Des Weiteren werden zwei CSS-Dateien („styles.css“ und „keyboard2.js.css“) und zwei Javascript-Dateien („keyboard2.js“ und „jquery.min.js“) eingebunden. Die beiden „keyboard“-Dateien werden für den klickbaren Keyboardersatz benötigt, die „jquery“-Datei für das Ein- und Ausblenden der Ausnahmen.

Weiters folgen selbst geschriebene Javascript-Funktionen, welche die Eingaben überprüfen und bei Falscheingabe einen Fehler ausgeben. Der nächste Javascript-Block beschäftigt sich mit dem Ein- und Ausblenden der, bereits erwähnten, Ausnahmen. Abschließend wird im `<head>`-Block der Titel festgelegt. All dies ist auf Abb. 82 ersichtlich.

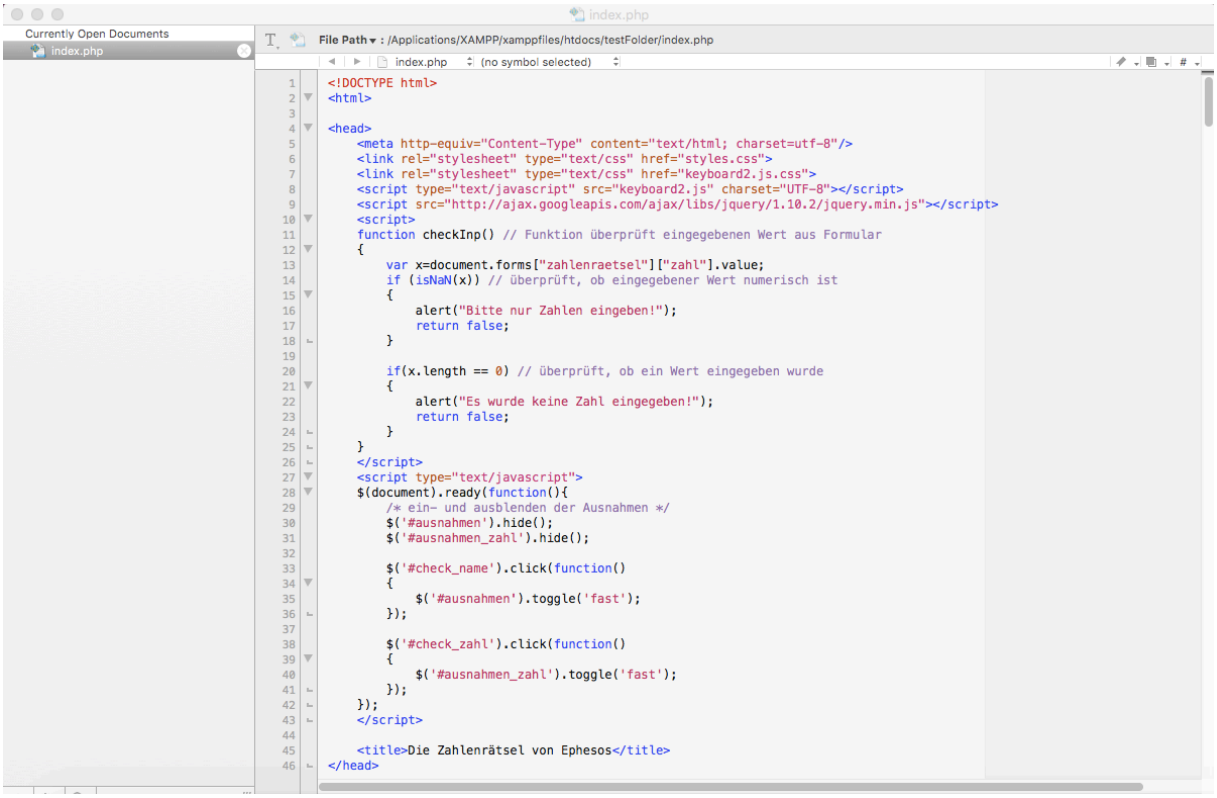


Abbildung 82 „index.php“ (Teil 1)

Im `<body>`-Block wird das Menü inkludiert und die Datenbankverbindung aufgebaut. Es folgt das Formular zur Eingabe der Zahl sowie die aufklappbaren Ausnahmen, welche mit Checkboxen ausgewählt werden können (siehe Abb. 83 und Abb. 84).

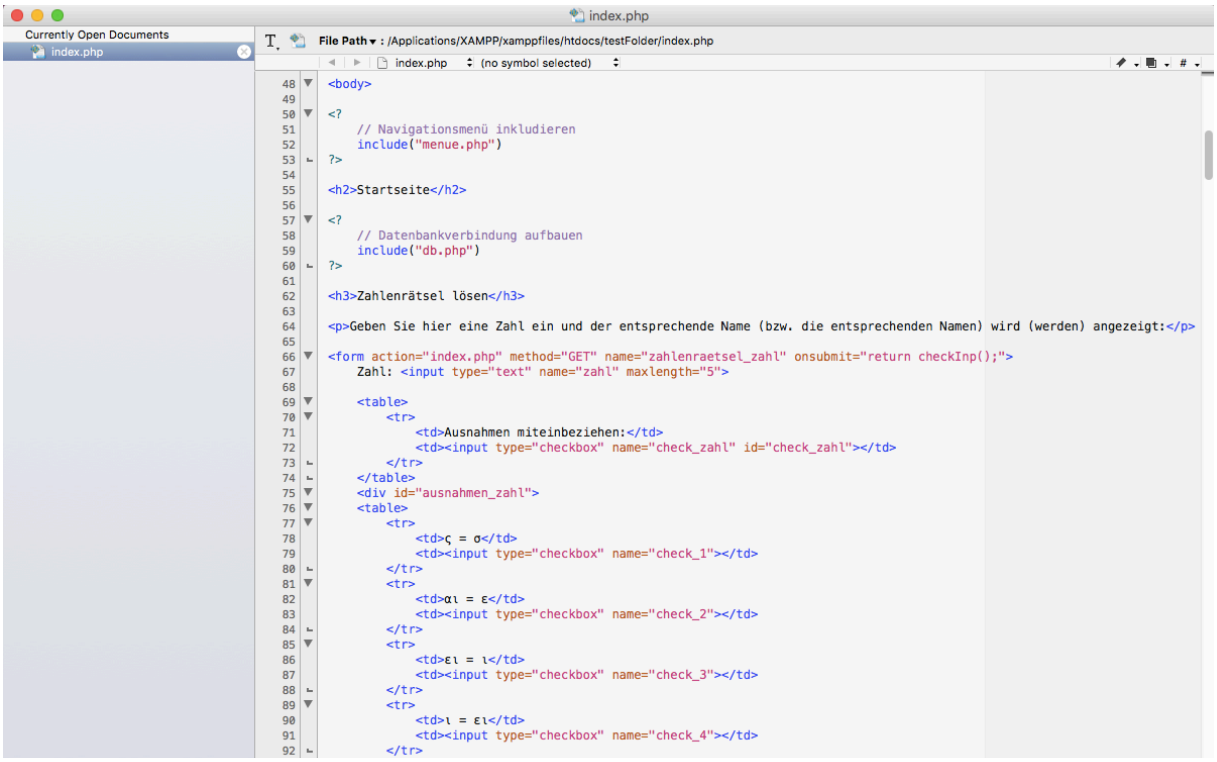


Abbildung 83 „index.php“ (Teil 2)

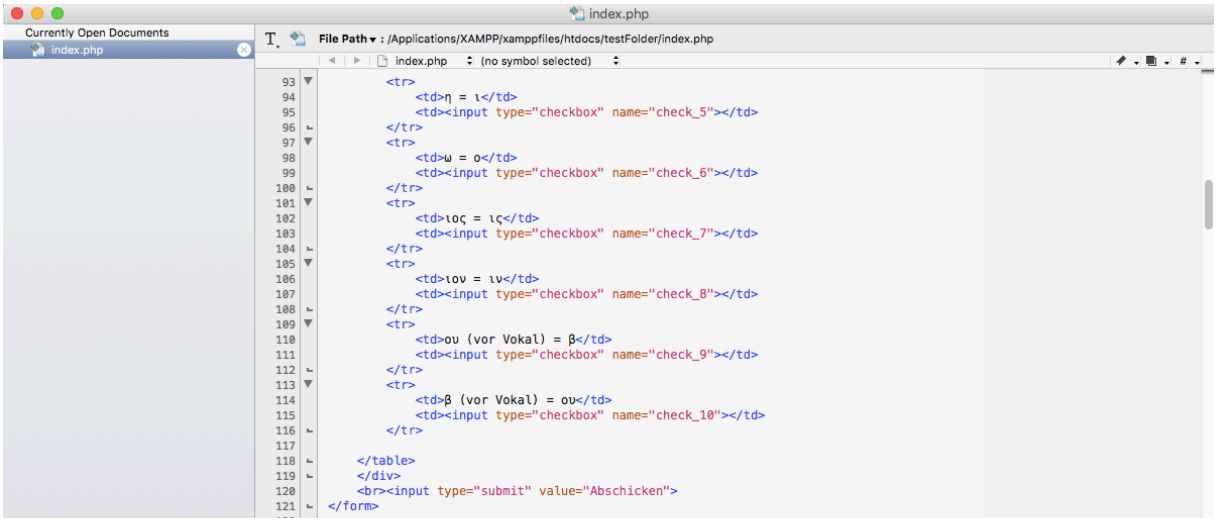


Abbildung 84 „index.php“ (Teil 3)

Wurde das Formular zur Eingabe der Zahl abgeschickt, so wird überprüft, ob eine Ausnahme ausgewählt worden ist. Ist dies der Fall, wird mittels Funktion „berechneAusnahme“ ein neuer Berechnungsvorgang gestartet (siehe Abb. 94-96 – am Ende der Datei „index.php“).

Soll der Name anhand der eingegebenen Zahl jedoch ohne Ausnahmen berechnet werden, so wird eine Datenbankabfrage abgesetzt, welche alle Namen ausgibt, die als „`$zahlenwert`“ die übergebene Zahl beinhalten. Sollte kein Name gefunden werden, so wird eine Fehlermeldung ausgegeben. Gibt es Ergebnisse, so werden diese in einer scrollbaren Box angezeigt.

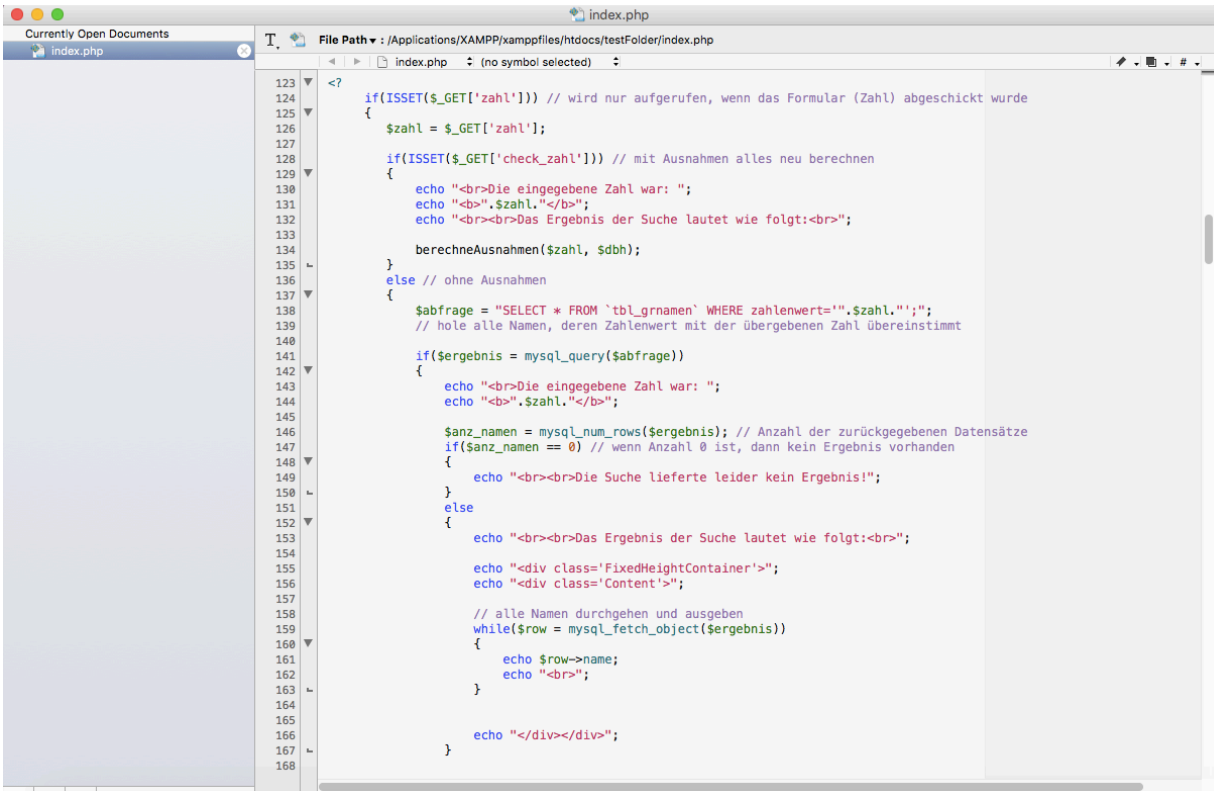


Abbildung 85 „index.php“ (Teil 4)

Es folgt nun ein zweites Formular, welches griechische Namen anhand der Umrechnungstabelle in eine Zahl umwandelt. Um eine Eingabe zu erleichtern, wird ein klickbarer Keyboardersatz (mit griechischen Buchstaben) eingebunden (als Klasse „keyboardInput“). Als Nächstes folgen, wie im soeben beschriebenen Codeabschnitt, die Ausnahmen, welche erneut durch Checkboxen ausgewählt werden und somit in die Berechnung miteinbezogen werden können (siehe Abb. 86 und Abb. 87).

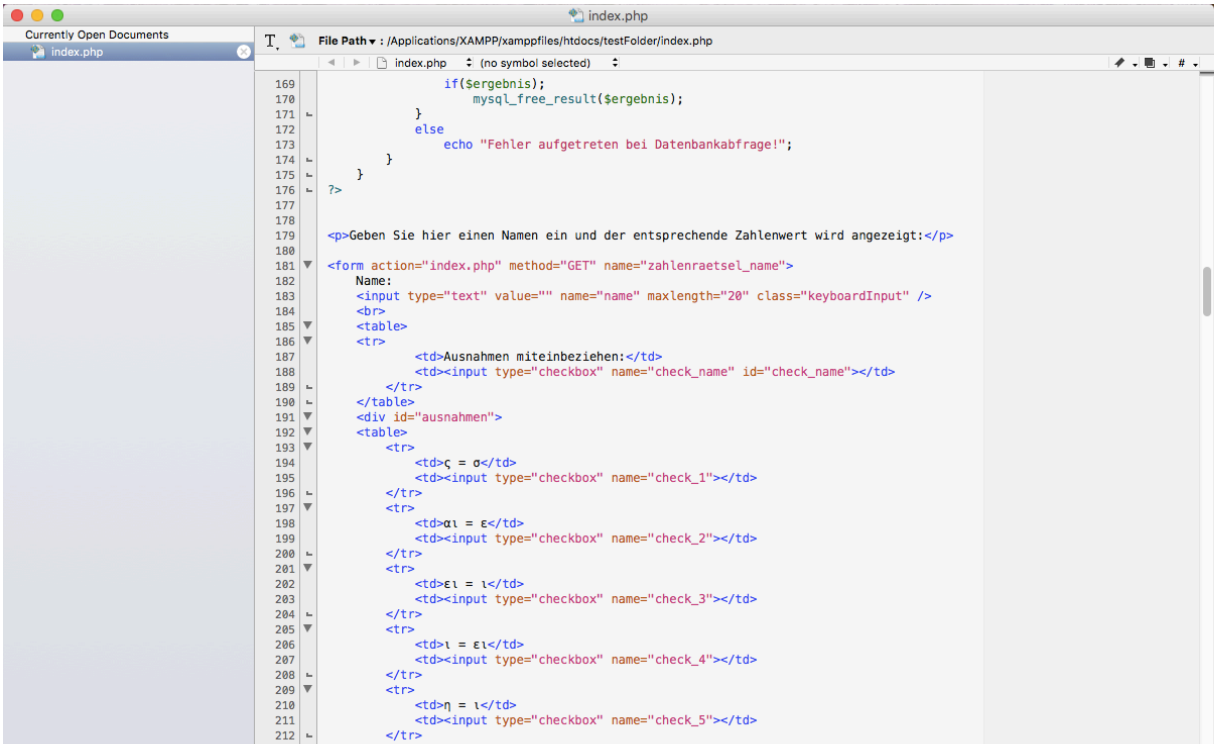


Abbildung 86 „index.php“ (Teil 5)

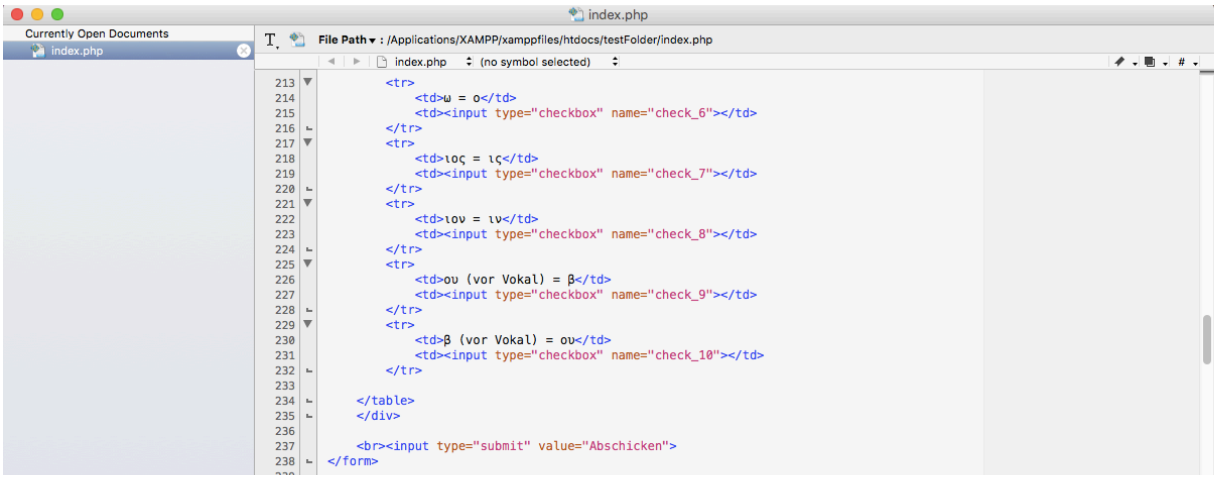


Abbildung 87 „index.php“ (Teil 6)

Wurde das Formular für die Namenseingabe abgeschickt, wird überprüft, ob Ausnahmen ausgewählt worden sind. Ist dies der Fall, wird sofort die Ausnahmeregelung auf den eingegebenen Namen angewandt (Buchstaben bzw. Buchstabenkombinationen werden

ersetzt). Anschließend wird der neue Name ausgegeben. Danach erfolgt die Zahlenwertberechnung: Dazu werden die einzelnen Buchstaben durchlaufen und der dazugehörige Zahlenwert gesucht und ausgegeben. Wird kein dazugehöriger Zahlenwert gefunden, so wird 0 ausgegeben.

Weiters wird überprüft, ob sich an der letzten Stelle des Namens ein Sigma befindet, denn dann handelt es sich um das so genannte „Schlusssigma“, welches einen anderen Wert als das „normale“ Sigma erhält (nämlich 6 statt 200). Diesen Unterschied kann die Datenbank nicht abbilden, jedoch wurde dies im Code abgefragt und behoben. Am Ende der Berechnung wird der gesamte Zahlenwert ausgegeben (siehe Abb. 88 und Abb. 89).

Will man keine Ausnahmen in die Berechnung miteinbeziehen, so wird die Berechnung wie oben beschrieben durchgeführt, jedoch wird der Schritt mit dem Ersetzen bestimmter Buchstaben („Ausnahmeregelung“) ausgelassen (siehe Abb. 90 und Abb. 91).

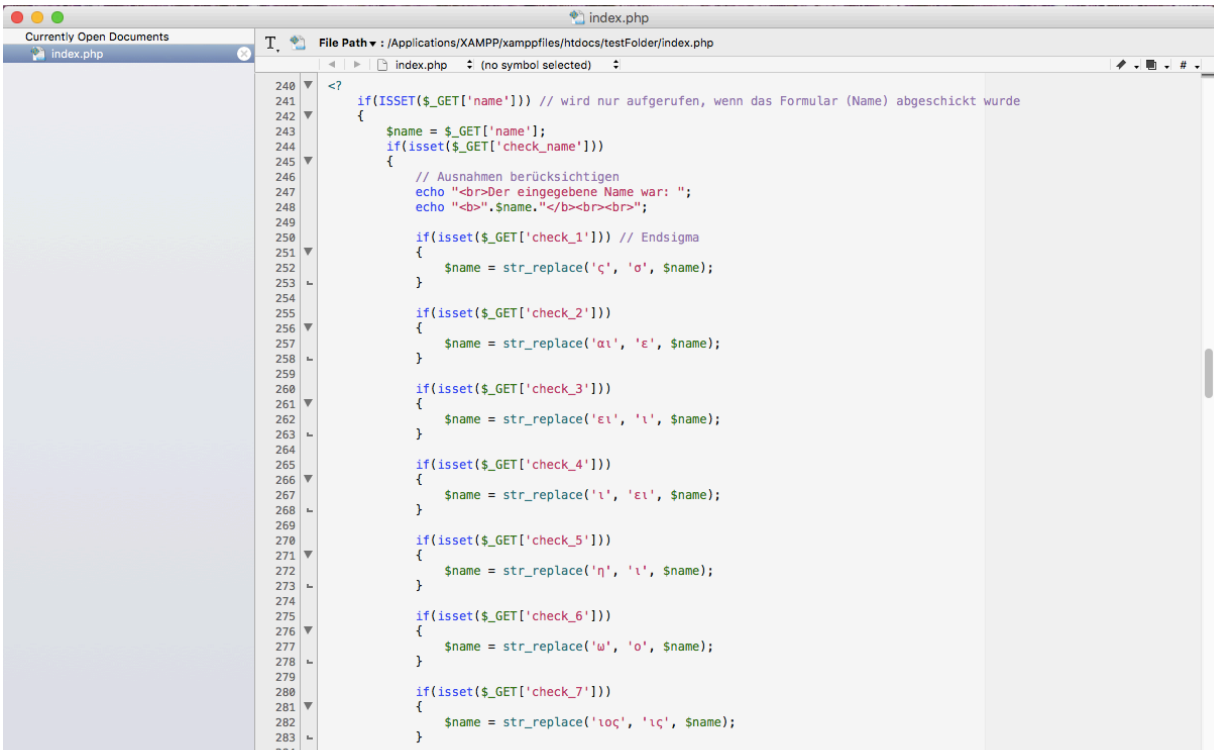
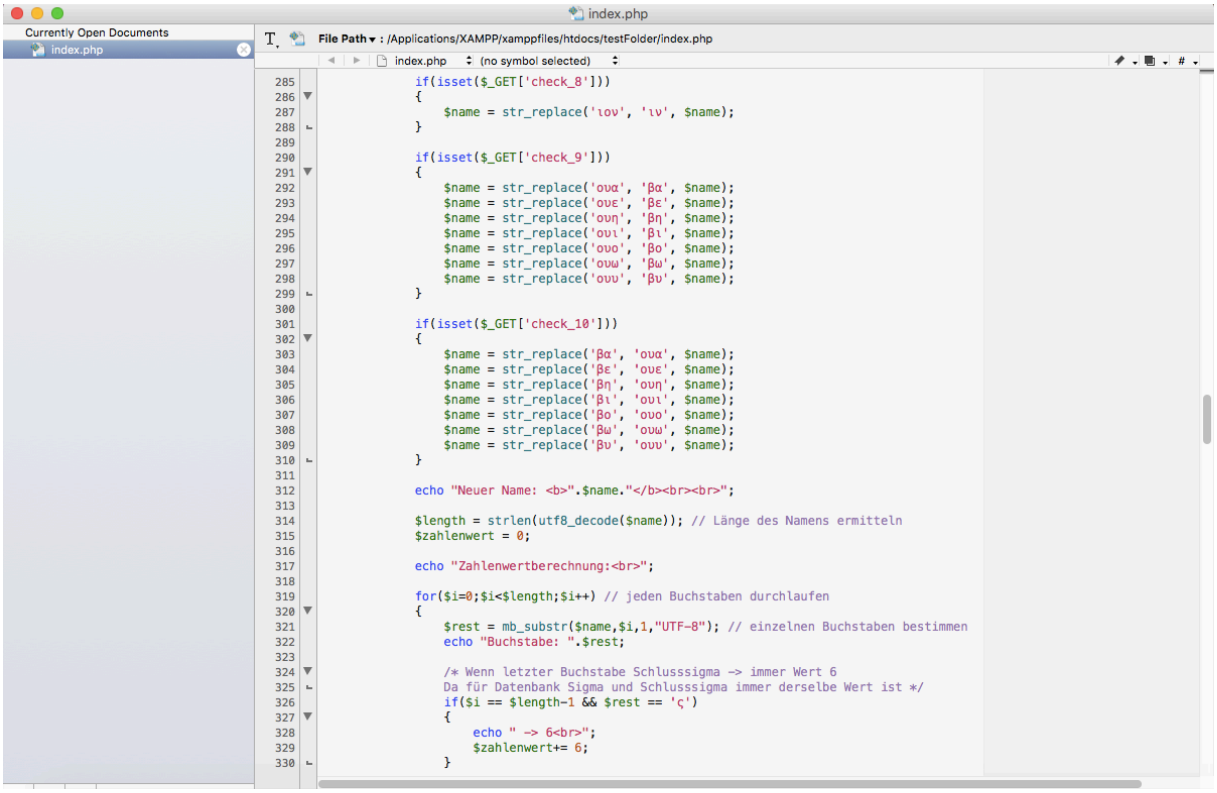
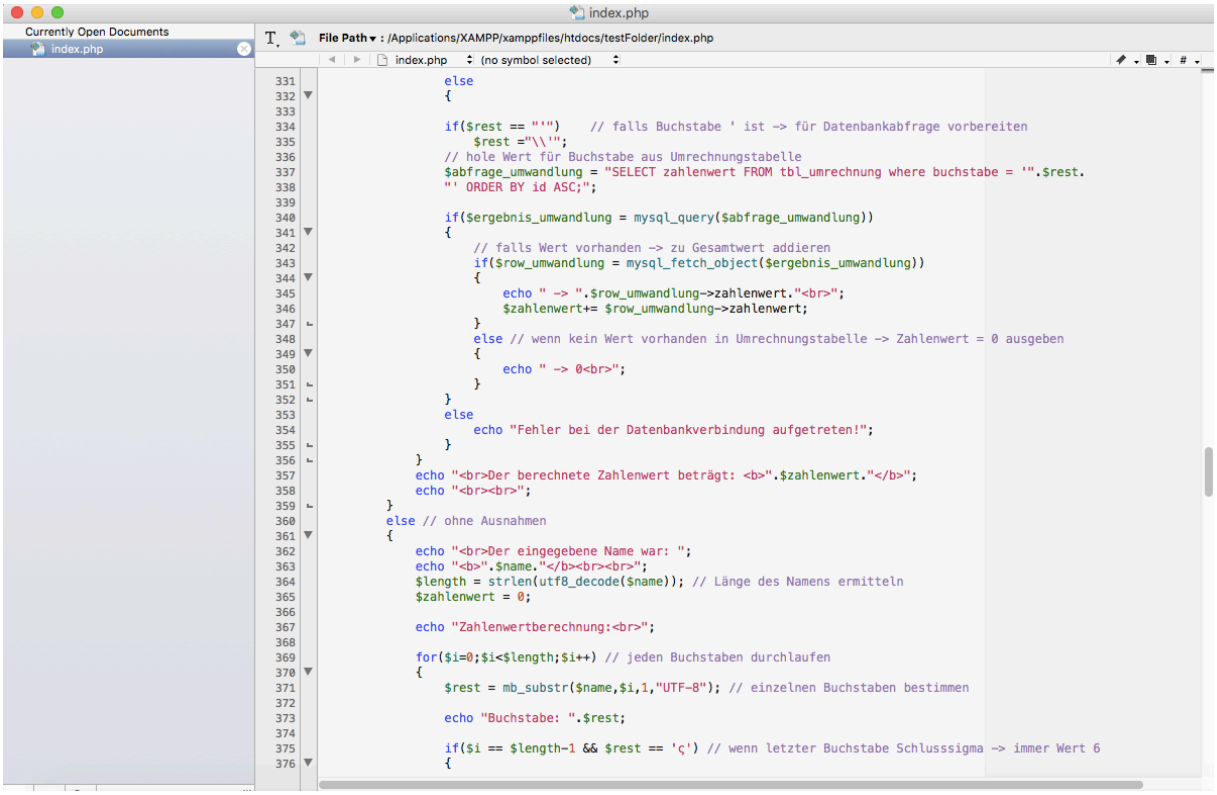


Abbildung 88 „index.php“ (Teil 7)




```
285             if(isset($_GET['check_8']))
286             {
287                 $name = str_replace('ιον', 'ιν', $name);
288             }
289
290             if(isset($_GET['check_9']))
291             {
292                 $name = str_replace('ουα', 'βα', $name);
293                 $name = str_replace('ουε', 'βε', $name);
294                 $name = str_replace('ουη', 'βη', $name);
295                 $name = str_replace('ουι', 'βι', $name);
296                 $name = str_replace('ουο', 'βο', $name);
297                 $name = str_replace('ουω', 'βω', $name);
298                 $name = str_replace('ουυ', 'βυ', $name);
299             }
300
301             if(isset($_GET['check_10']))
302             {
303                 $name = str_replace('βα', 'ουα', $name);
304                 $name = str_replace('βε', 'ουε', $name);
305                 $name = str_replace('βη', 'ουη', $name);
306                 $name = str_replace('βι', 'ουι', $name);
307                 $name = str_replace('βο', 'ουο', $name);
308                 $name = str_replace('βω', 'ουω', $name);
309                 $name = str_replace('βυ', 'ουυ', $name);
310             }
311
312             echo "Neuer Name: <b>".$name."</b><br><br>";
313
314             $length = strlen(utf8_decode($name)); // Länge des Namens ermitteln
315             $zahlenwert = 0;
316
317             echo "Zahlenwertberechnung:<br>";
318
319             for($i=0;$i<$length;$i++) // jeden Buchstaben durchlaufen
320             {
321                 $rest = mb_substr($name,$i,1,"UTF-8"); // einzelnen Buchstaben bestimmen
322                 echo "Buchstabe: ".$rest;
323
324                 /* Wenn letzter Buchstabe Schlusssigma -> immer Wert 6
325                 Da für Datenbank Sigma und Schlusssigma immer derselbe Wert ist */
326                 if($i == $length-1 && $rest == 'ς')
327                 {
328                     echo " -> 6<br>";
329                     $zahlenwert+= 6;
330                 }
```

Abbildung 89 „index.php“ (Teil 8)




```
331                 else
332                 {
333
334                 if($rest == "'")    // falls Buchstabe ' ist -> für Datenbankabfrage vorbereiten
335                     $rest ="\\'";
336                 // hole Wert für Buchstabe aus Umrechnungstabelle
337                 $abfrage_umwandlung = "SELECT zahlenwert FROM tbl_umrechnung where buchstabe = '".$rest.
338                 "' ORDER BY id ASC;";
339
340                 if($ergebnis_umwandlung = mysql_query($abfrage_umwandlung))
341                 {
342                     // falls Wert vorhanden -> zu Gesamtwert addieren
343                     if($row_umwandlung = mysql_fetch_object($ergebnis_umwandlung))
344                     {
345                         echo " -> ".$row_umwandlung->zahlenwert."<br>";
346                         $zahlenwert+= $row_umwandlung->zahlenwert;
347                     }
348                     else // wenn kein Wert vorhanden in Umrechnungstabelle -> Zahlenwert = 0 ausgeben
349                     {
350                         echo " -> 0<br>";
351                     }
352                 }
353                 else
354                     echo "Fehler bei der Datenbankverbindung aufgetreten!";
355                 }
356             }
357             echo "<br>Der berechnete Zahlenwert beträgt: <b>".$zahlenwert."</b>";
358             echo "<br><br>";
359         }
360         else // ohne Ausnahmen
361         {
362             echo "<br>Der eingegebene Name war: ";
363             echo "<b>".$name."</b><br><br>";
364             $length = strlen(utf8_decode($name)); // Länge des Namens ermitteln
365             $zahlenwert = 0;
366
367             echo "Zahlenwertberechnung:<br>";
368
369             for($i=0;$i<$length;$i++) // jeden Buchstaben durchlaufen
370             {
371                 $rest = mb_substr($name,$i,1,"UTF-8"); // einzelnen Buchstaben bestimmen
372
373                 echo "Buchstabe: ".$rest;
374
375                 if($i == $length-1 && $rest == 'ς') // wenn letzter Buchstabe Schlusssigma -> immer Wert 6
376                 {
```

Abbildung 90 „index.php“ (Teil 9)

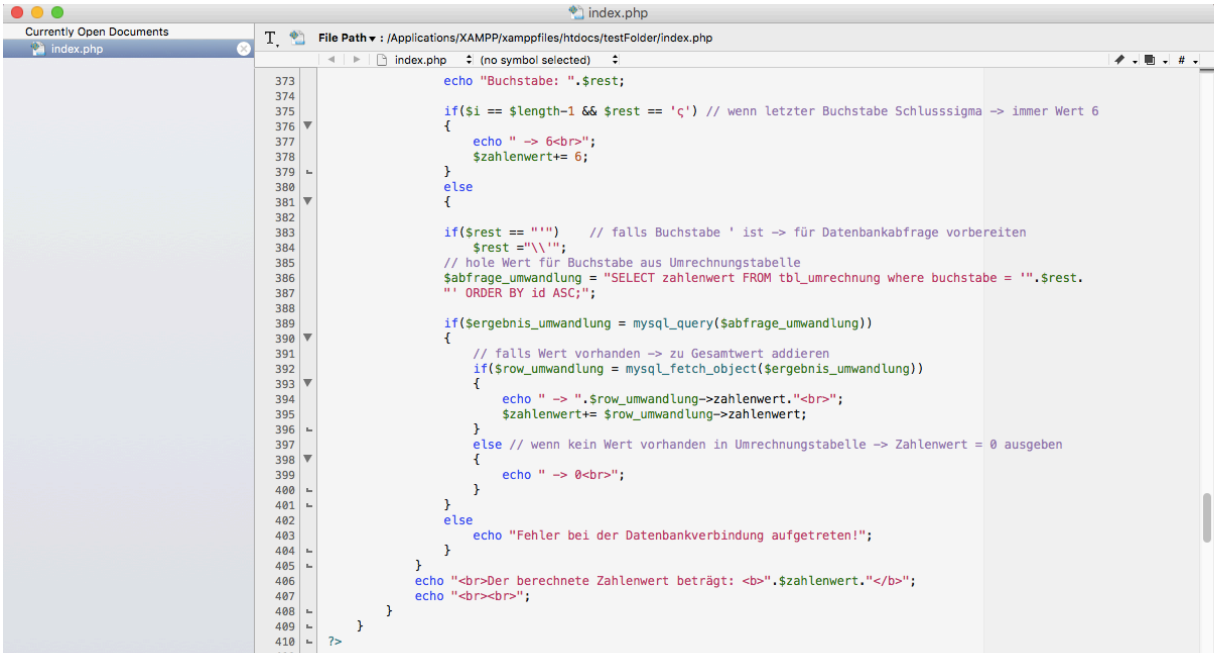


Abbildung 91 „index.php“ (Teil 10)

Nach den Formularen werden die bisher gelösten Zahlenrätsel angezeigt, welche in der Datenbank gespeichert sind. Dazu werden alle Einträge der Tabelle „tbl_raetsel“ ausgegeben. Da manche Rätsel eine Ausnahme enthalten können, wurde ein zusätzliches Attribut „ausnahme_sigma“ vergeben, welches der Funktion „berechneAusnahmen“ übergeben wird. Wenn keine Ausnahme berechnet wird, wird der Zahlenwert in der Tabelle „tbl_grnamen“ gesucht und ausgegeben (siehe Abb. 92 und Abb. 93).

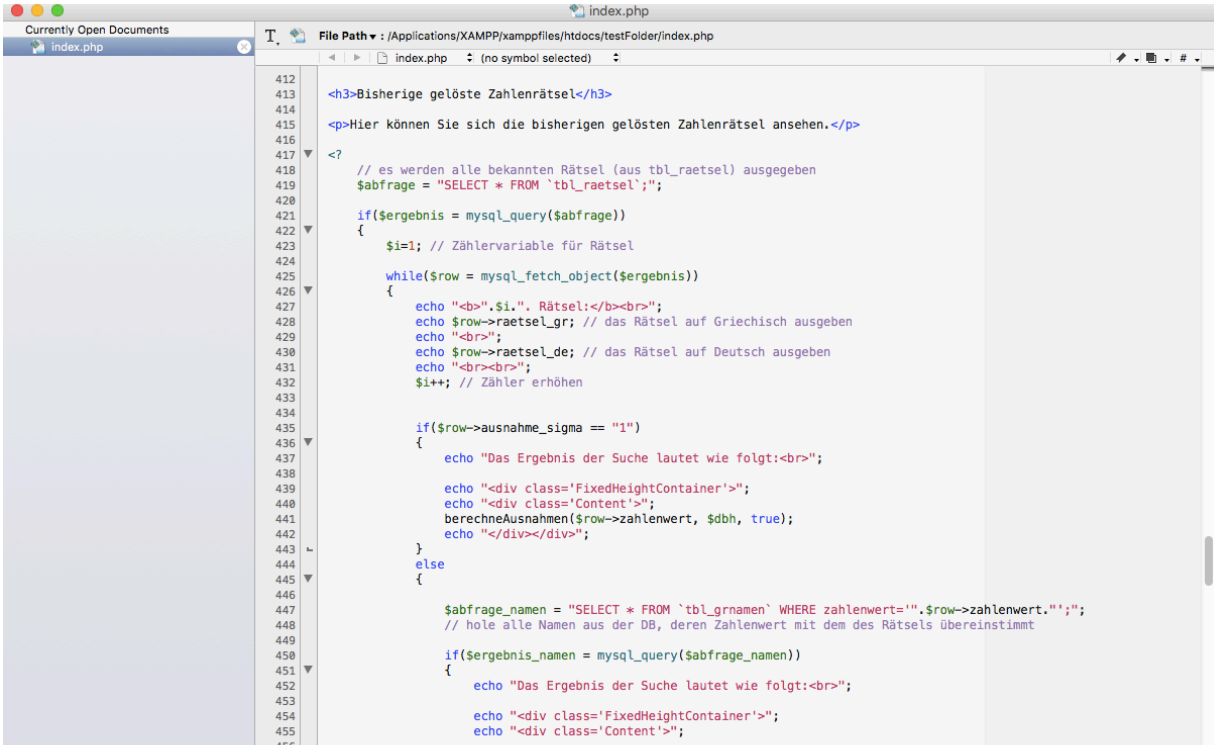


Abbildung 92 „index.php“ (Teil 11)

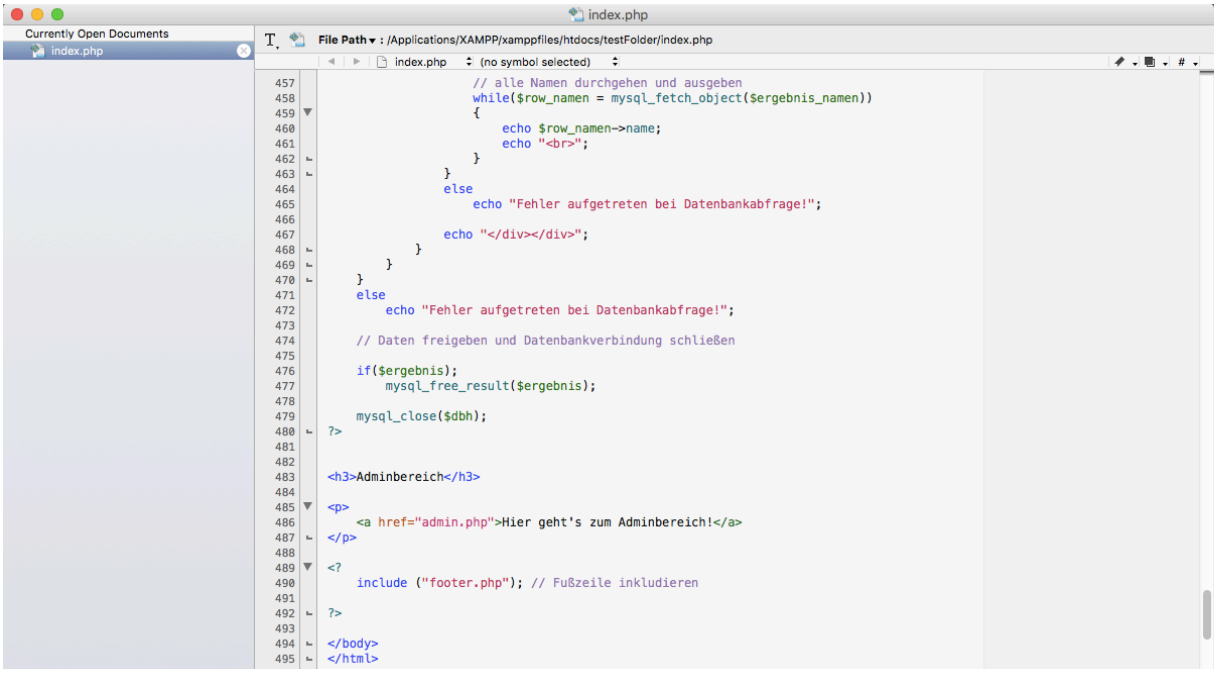


Abbildung 93 „index.php“ (Teil 12)

Am Ende der Datei wird die Funktion „berechneAusnahmen“ definiert, welche drei Übergabeparameter besitzt – die Zahl, die Datenbankverbindung und ob das Ausnahme-Sigma (wie soeben angesprochen) berücksichtigt werden soll. Um eine bessere Laufzeit zu gewähren, wird die Umrechnungstabelle in ein Array gespeichert (Vorteil: keine unnötig vielen Abfragen für jeden einzelnen Buchstaben – wie dies in früheren Versionen gehandhabt wurde).

Als Nächstes werden alle Namen aus der Datenbank geholt und Buchstaben infolge der Ausnahmeregelung ersetzt (also wenn Ausnahmen ausgewählt wurden – das trifft auch auf das Schlusssigma zu, welches übergeben werden kann).

Dann wird für jeden Namen ein Zahlenwert berechnet und mit dem übergebenen verglichen. Stimmen die beiden Werte überein, wird der Name ausgegeben und zählt somit zu den Ergebnissen (siehe Abb. 94, Abb. 95 und Abb. 96).

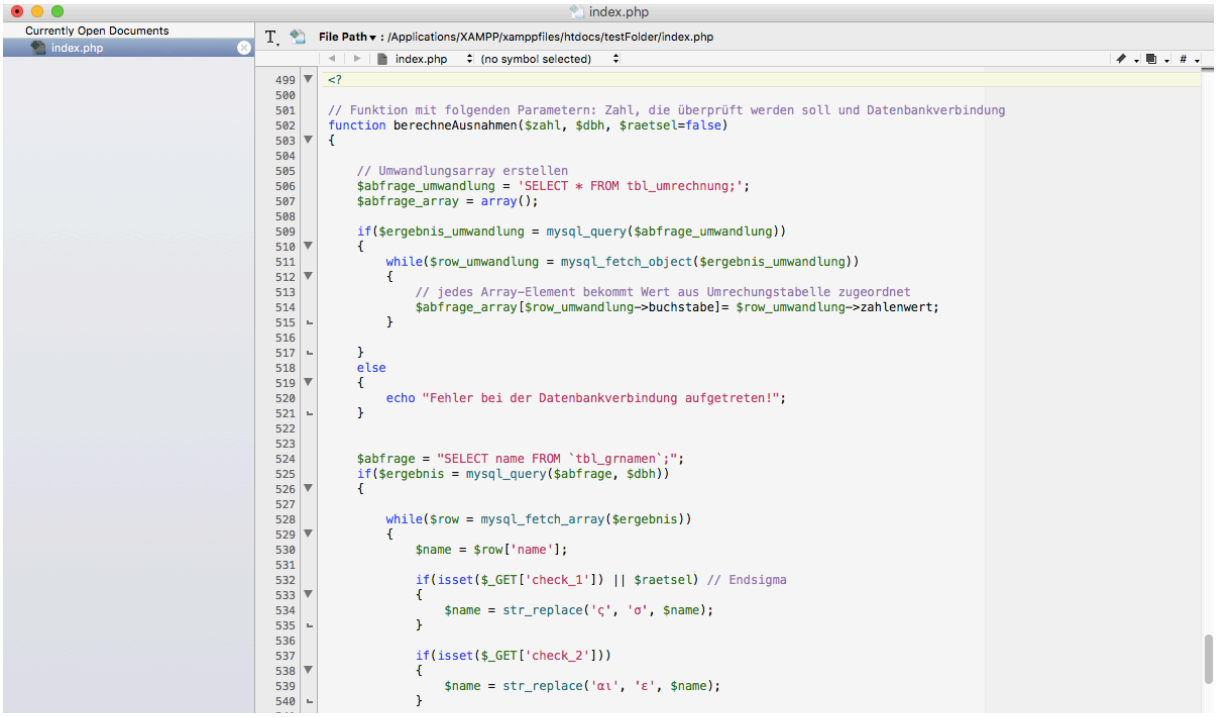


Abbildung 94 „index.php“ (Teil 13)

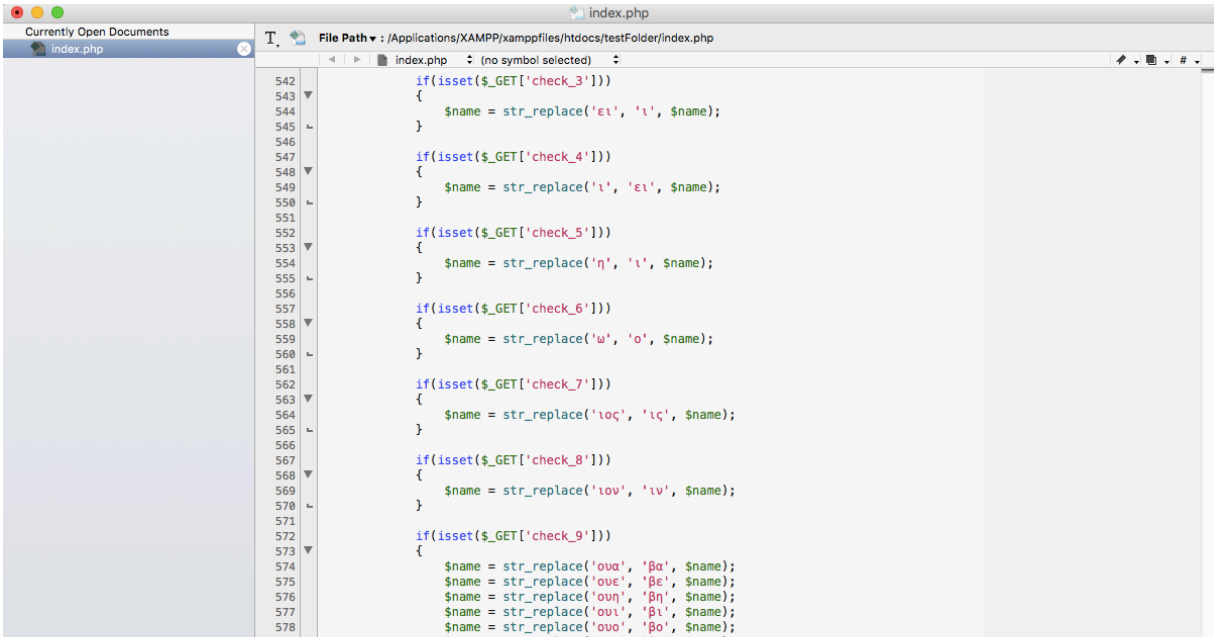


Abbildung 95 „index.php“ (Teil 14)

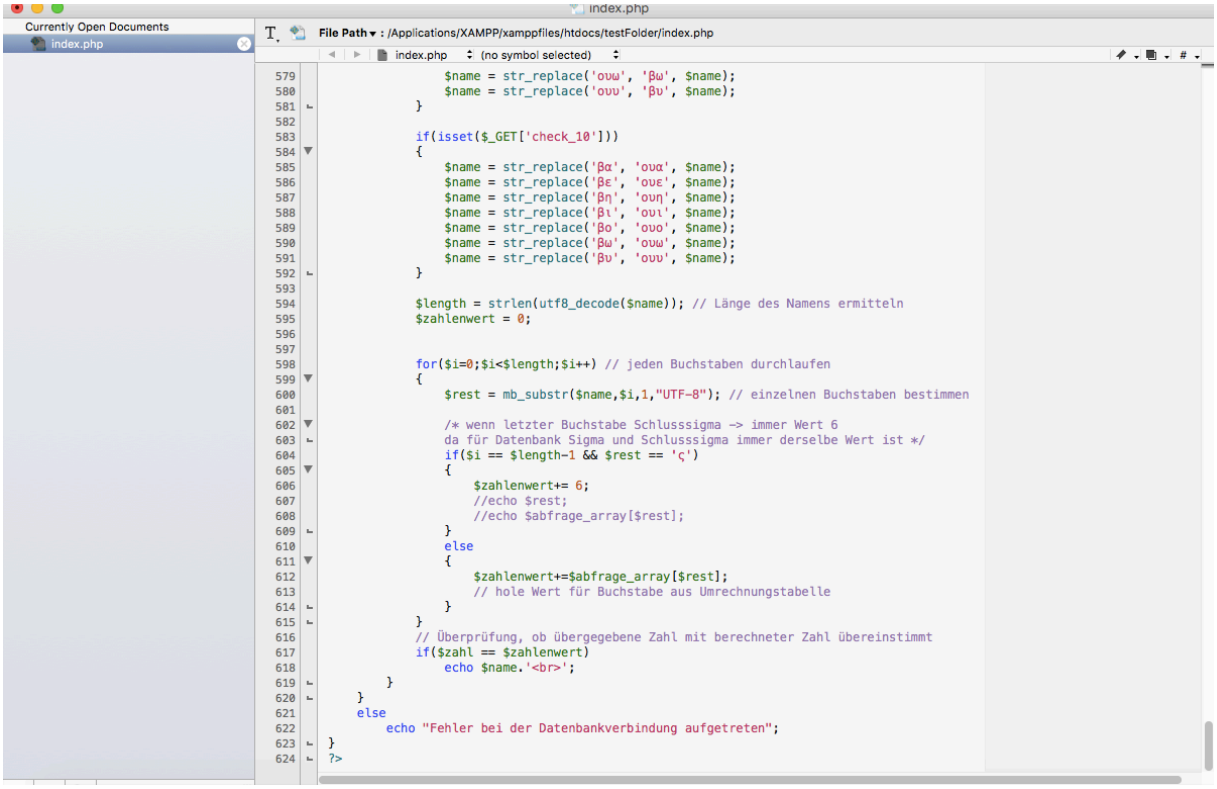


Abbildung 96 „index.php“ (Teil 15)

Datei „menue.php“

Diese Datei ist zur Einbettung in andere Dateien gedacht. Sie gibt die Überschrift des Web Interfaces und die obere Menüstruktur aus und ist des Weiteren auch für das Logout zuständig.

Abbildung 97 Code „menue.php“ (Teil 1)

Im Kommentar wird (wie bereits beschrieben) der Name der Datei und die Autorin der Datei definiert. Mit `session_start()` wird eine neue Session erstellt oder eine bereits vorhandene fortgesetzt. Wurde der Logout-Link gedrückt, so wird die vorhandene Session mittels `session_destroy()` gelöscht und automatisch auf die „loginindex.php“-Seite weitergeleitet. Ist keine Session gesetzt, so wird ebenfalls auf die „loginindex.php“-Seite weitergeleitet.

```php
    // Menüeinträge mit dazugehöriger PHP-Datei in Array speichern
    $menue = array(
        "Startseite" => "index.php",
        "Über dieses Projekt" => "ueberprojekt.php",
        "Über die Zahlenrätsel" => "ueberraetsel.php",
        "Impressum" => "impressum.php",
    );
?>
```

Abbildung 10 Code „menue.php“ (Teil 2)

Im Array `$menue` sind alle Menüeinträge und die Seiten, auf die sie verweisen, gespeichert. So ist beispielsweise im Array-Index „Startseite“ die Seite „index.php“ gespeichert. Auf die folgenden drei Zeilen trifft dasselbe zu (siehe Abb. 98). Hier können jederzeit beliebig viele Menüeinträge hinzugefügt oder gelöscht werden, welche dann auf jeder Seite angezeigt werden.

```php
<h1>Die Zahlenrätsel von Ephesos</h1>

<p align="right">
    <a href="index.php?logout">Logout</a>
</p>
```

Abbildung 99 Code „menue.php“ (Teil 3)

Im `<h1>`-Tag wird die Überschrift des Projekts festgelegt und darunter, mit Textausrichtung rechtsbündig, der Logout-Button angezeigt (siehe Abb. 99).

```php
<table width="100%">
    <tr>
        <?
            foreach($menue as $entry => $datei)
            // Menüeinträge einzeln durchgehen und anzeigen
            {
                if ($datei == basename($_SERVER['PHP_SELF']))
                // gerade ausgewählten Menüeintrag kennzeichnen
                {
                    echo "<td align=\"center\" bgcolor=\"#8B5A2B\" id=\"menue1\">";
                    echo "<a href=\"$datei\" id=\"menue1\">$entry</a>";
                }
                else
                {
                    echo "<td align=\"center\" bgcolor=\"#CD8500\" id=\"menue2\">";
                    echo "<a href=\"$datei\">$entry</a>";
                    echo "</td>";
                }
            }
        ?>
    </tr>
</table>
```

Abbildung 100 Code „menue.php“ (Teil 4)

In der darauffolgenden Tabelle werden dann alle Menüeinträge aus dem Array mittels `foreach` ausgelesen und das Menü aufgebaut. Der Menühintergrund (und die Farbe des Namens) der aktuellen Seite ist anders eingefärbt als der der übrigen Menülinks. Dies wird durch die `if`-Abfrage `($datei == basename($_SERVER['PHP_SELF']))` bewerkstelligt.

Datei „ueberprojekt.php“

In dieser Datei wird Diverses über das Projekt berichtet, wie beispielsweise wann es entstanden ist und wer die Betreuer dieser Diplomarbeit sind.

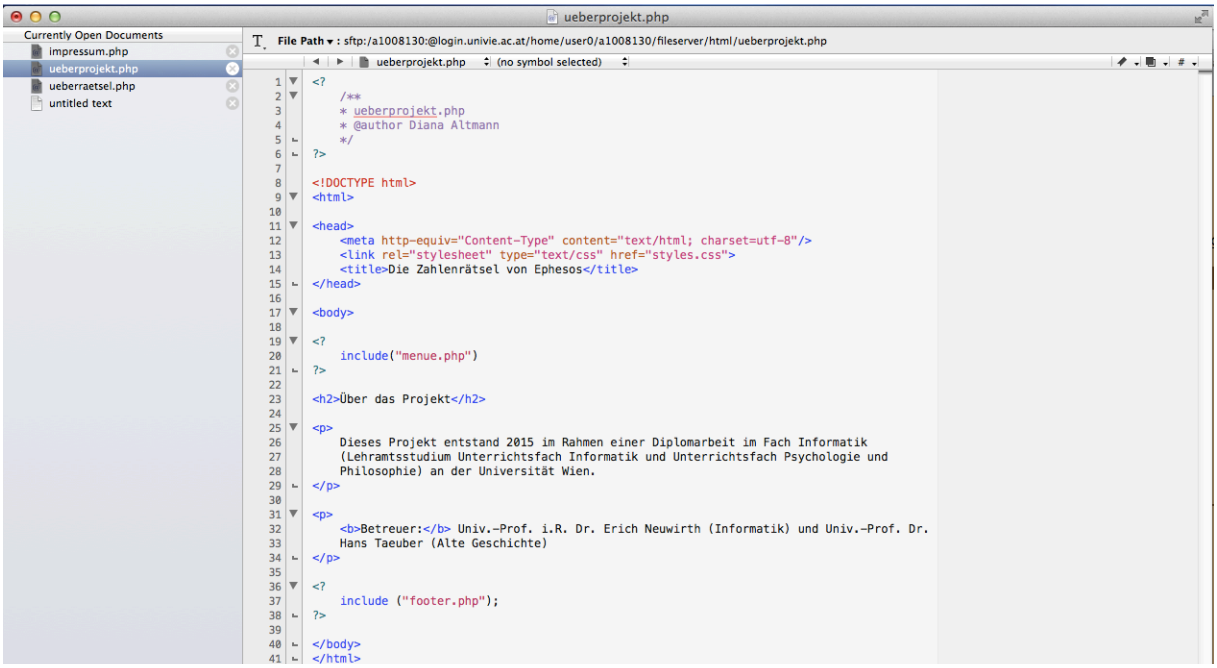


Abbildung 101 Code „ueberprojekt.php“

Der Code beginnt erneut mit einem Kommentar (Inhalt sind Name der Datei und Name der Autorin der Datei). Der Doctype wird mittels `<!DOCTYPE html>` definiert und die Grundstruktur einer HTML-Seite eingebettet (wie bereits erläutert). Im `<head>`-Bereich befindet sich der Meta-Tag, welcher hier UTF-8 als Zeichencodierung festlegt. Die folgende Codezeile bindet die CSS-Datei namens „styles.css“ ein.

Im `<body>`-Bereich wird die Überschrift „Über die Rätsel von Ephesos“ ausgegeben und danach zwei Absätze mit Text in `<p>`-Tags definiert. Am Ende erfolgt das Einbinden und die Ausgabe der Fußzeile („footer.php“) und anschließend wird der `<body>`-Bereich als auch der `<html>`-Bereich geschlossen (mittels `</body>` und `</html>`).

Datei „ueberraetsel.php“

Diese Datei beschreibt die Rätsel von Ephesos im Allgemeinen. Der Aufbau dieser Seite verhält sich prinzipiell gleich wie der von der Seite „ueberprojekt.php“.

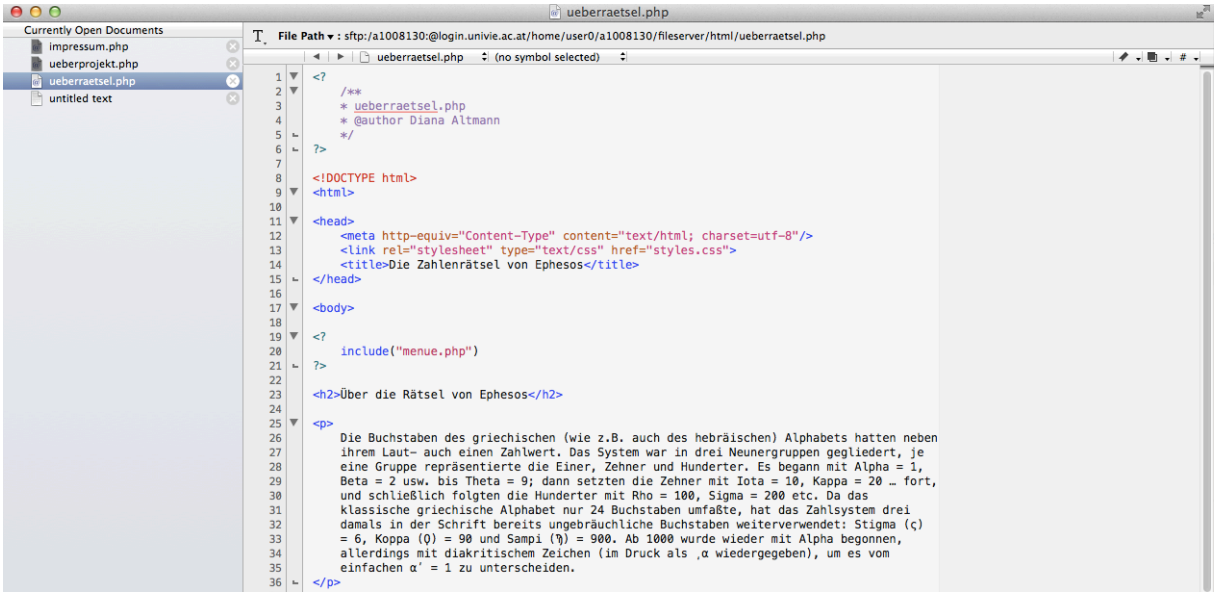


Abbildung 102 Code „ueberraetsel.php“ (Teil 1)

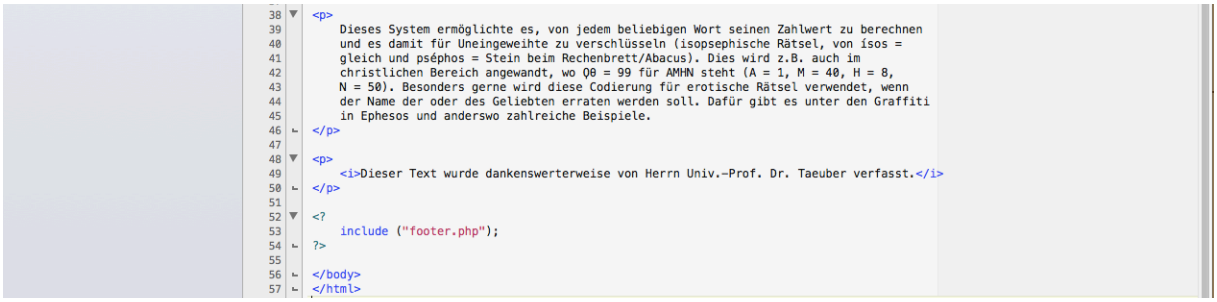


Abbildung 103 Code „ueberraetsel.php“ (Teil 2)

Datei „impressum.php“

Die Datei beinhaltet Informationen zur Autorin dieses Web Interfaces sowie zur Kontaktaufnahme. Sie ist im Wesentlichen ebenfalls gleich aufgebaut wie die Datei „ueberprojekt.php“, mit dem einzigen Unterschied, dass mit Hilfe des „mailto“-Befehls auf Klick eine Mail an meine hinterlegte E-Mail Adresse gesendet werden kann.

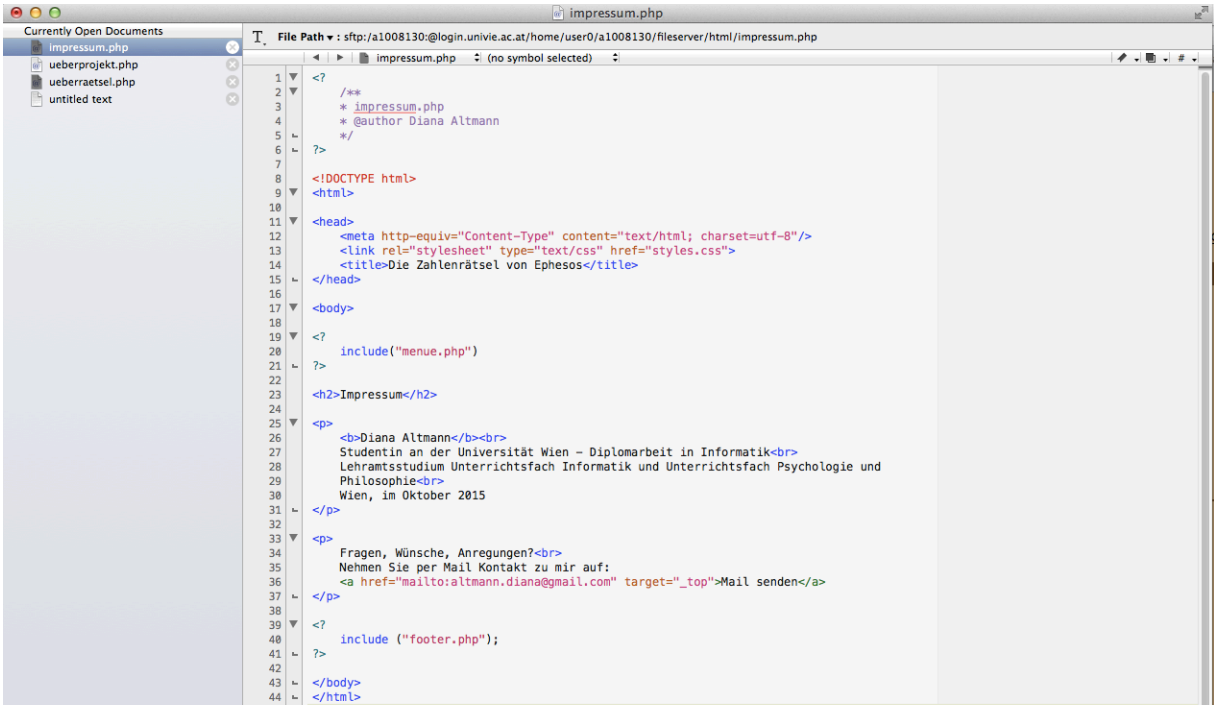


Abbildung 104 Code „impressum.php“

Datei „footer.php“

Diese Datei ist zur Einbettung in andere Dateien gedacht (wie bereits erwähnt). Sie gibt Informationen über das Projekt, die Autorin und das Datum der letzten Änderung aus.

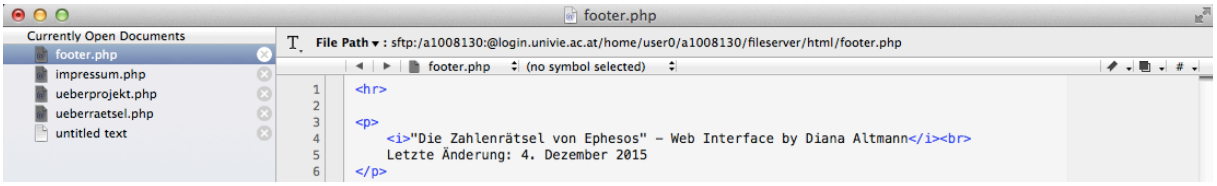


Abbildung 105 Code „footer.php“

Datei „styles.css“

In dieser Datei werden layoutspezifische Eigenschaften festgelegt. So wurden beispielsweise für die unterschiedlichen Überschriften (`<h1>`, `<h2>`, `<h3>`) verschiedene Schriftgrößen und Schriftfarben ausgewählt. Für Links wurden zwei unterschiedliche Farben vergeben – einerseits die Darstellungsfarbe (`a`) und andererseits die Farbe, die der Link erhält, wenn man mit dem Mauszeiger darüberfährt (`a hover`).

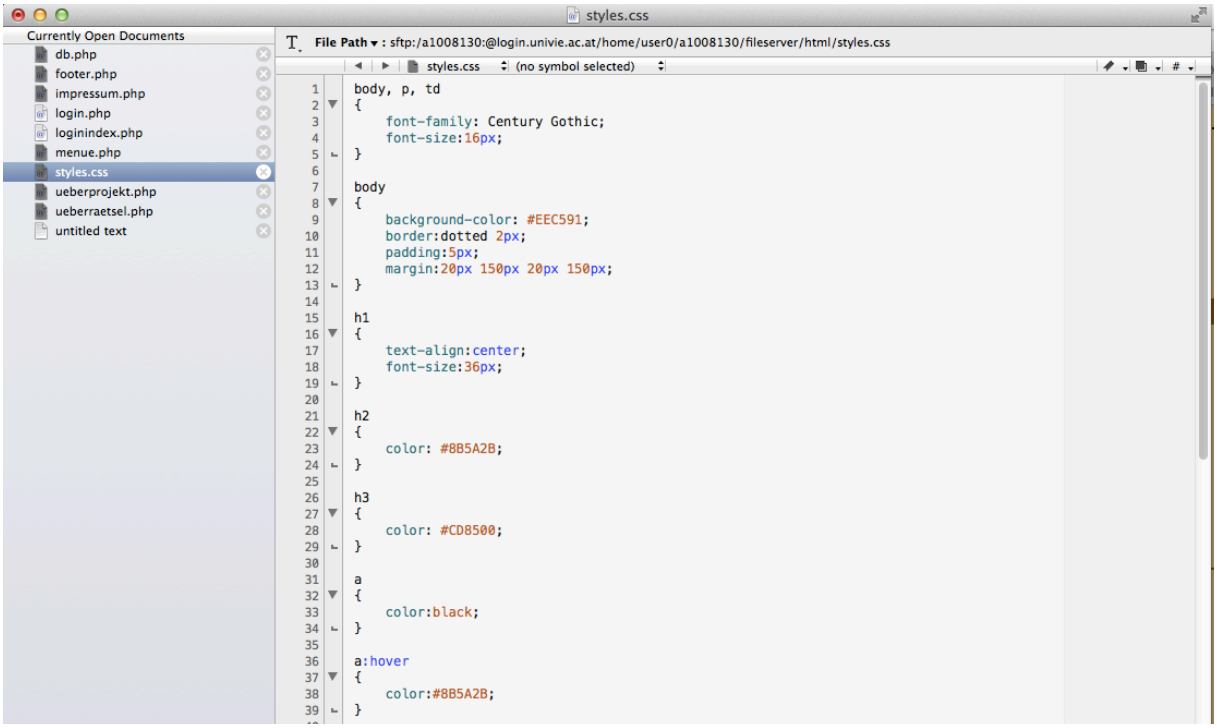


Abbildung 106 Code „styles.css“ (Teil 1)

Des Weiteren wurden die Eigenschaften der Trennlinie (`<hr>`) festgelegt und die von diversen Menüstrukturen („`#menue1`“ aktuell ausgewählte Seite im Menü, „`#menue2`“ andere Seiten im Menü). Am Ende der Datei wird noch das Layout der scrollbaren Boxen, welche sich auf der Startseite befinden, festgelegt (unter „`.FixedHeightContainer`“ und „`.Content`“) – siehe Abb. 107 und Abb. 108.

Abbildung 107 Code „styles.css“ (Teil 2)

Abbildung 108 Code „styles.css“ (Teil 3)

Datei „login.php“

Diese Seite ist dazu da, um dem Adminbereich des „Zahlenrätsel von Ephesos“-Web Interfaces ein zwingendes Login vorzuschalten, wie dies auch bei dem Passwortschutz für die „index.php“-Seite der Fall ist. Daher verhält sich der Aufbau dieser Seite prinzipiell gleich wie der von der Seite „loginindex.php“. Eine Ausnahme ist die Weiterleitung auf „admin.php“ nach korrekter Eingabe der Logindaten.

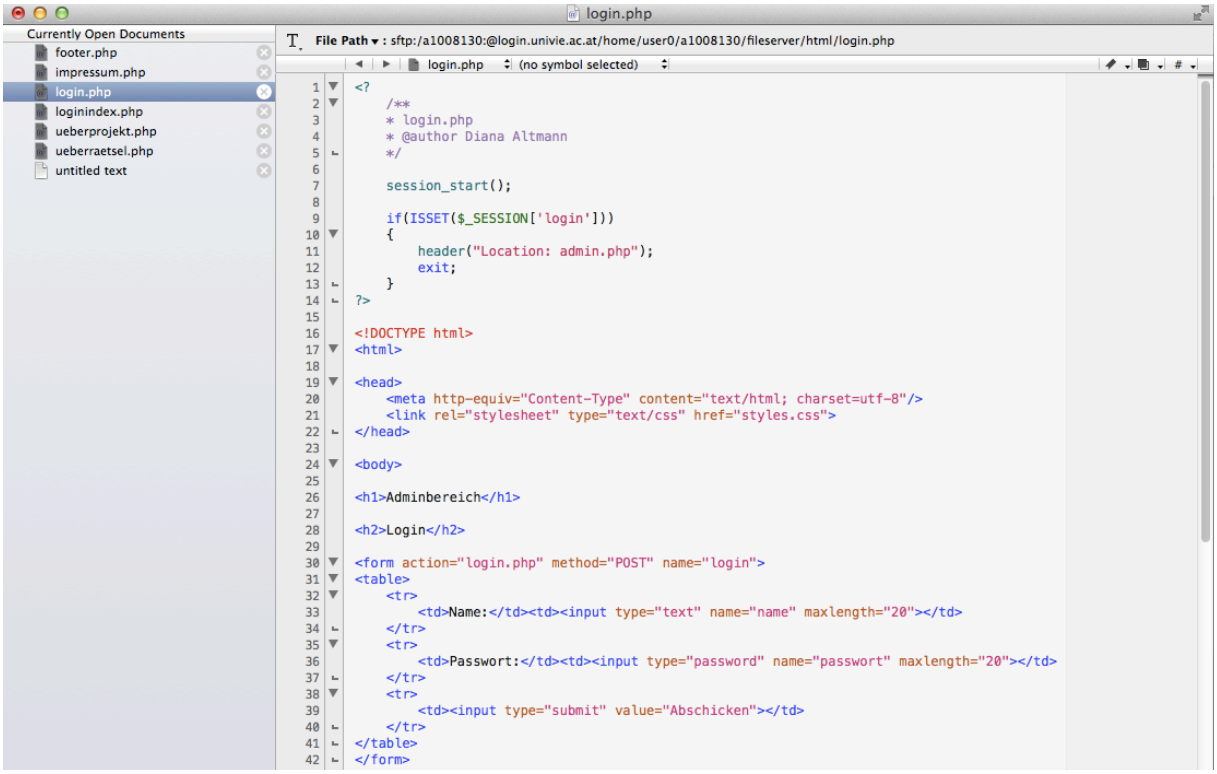


Abbildung 109 Code „login.php“ (Teil 1)

Abbildung 110 Code „login.php“ (Teil 2)

Datei „admin.php“

Diese Seite ist die Startseite des Adminbereichs, auf welcher erste Tests mit der Datenbank durchgeführt wurden. Der HTML-Teil dieser Datei ist gleich wie bei den anderen Seiten, nur dass hier das Adminmenü („adminmenue.php“) eingebunden wird.

Bezogen auf PHP wird hier in Zeile 22 die ausgelagerte Datenbankverbindung eingebunden und aufgebaut. Anschließend wird eine Abfrage durchgeführt („zeige die ersten 200 griechischen Namen aus der Tabelle tbl_grnamen an inklusive Buchstabenberechnung“).

admin.php

Currently Open Documents
admin.php
admin2.php
adminmenue.php
menue.php
untitled text

File Path ▾ : sftp://a1008130:@login.univie.ac.at/home/user0/a1008130/fileserver/html/admin.php

admin.php (no symbol selected)

```
<!DOCTYPE html>
<html>

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>

<body>

<?
    // Menü inkludieren
    include("adminmenue.php")
?>

<h2>Zahlenwerte für Einträge der Namenstabelle berechnen</h2>

<h3>Für die ersten 200 Datensätze:</h3>

<?
    // Datenbankverbindung herstellen
    include("db.php");

    // hole die ersten 200 griechischen Namen aus Datenbank
    $abfrage = "SELECT * FROM `tbl_grnamen` LIMIT 200";
    if($ergebnis = mysql_query($abfrage, $dbh))
    {
        while($row = mysql_fetch_object($ergebnis))
        {
            echo "<b>".$row->name."</b>";
            echo "<br>";
            $length = strlen(utf8_decode($row->name)); // Länge des Namens ermitteln
            $zahlenwert = 0;
```

Abbildung 111 „admin.php“ (Teil 1)

admin.php

Currently Open Documents
admin.php
admin2.php
adminmenue.php
menue.php
untitled text

File Path ▾ : sftp://a1008130:@login.univie.ac.at/home/user0/a1008130/fileserver/html/admin.php

admin.php (no symbol selected)

```
            for($i=0;$i<$length;$i++) // jeden Buchstaben durchlaufen
            {
                $rest = mb_substr($row->name,$i,1,"UTF-8"); // einzelnen Buchstaben bestimmen

                echo "Buchstabe: ".$rest;
                if($rest == "'")    // falls Buchstabe ' ist -> für Datenbankabfrage vorbereiten
                    $rest ="\\'";
                // hole Wert für Buchstabe aus Umrechnungstabelle
                $abfrage_umwandlung = "SELECT zahlenwert FROM tbl_umrechnung where buchstabe = '".$rest."';";

                if($ergebnis_umwandlung = mysql_query($abfrage_umwandlung))
                {
                    // falls Wert vorhanden -> zu Gesamtwert addieren
                    if($row_umwandlung = mysql_fetch_object($ergebnis_umwandlung))
                    {
                        echo " -> ".$row_umwandlung->zahlenwert."<br>";
                        $zahlenwert+= $row_umwandlung->zahlenwert;
                    }
                    else // wenn kein Wert vorhanden in Umrechnungstabelle -> Zahlenwert = 0 ausgeben
                    {
                        echo " -> 0<br>";
                    }
                }
                else
                    echo "Fehler bei der Datenbankverbindung aufgetreten!";
            }
            echo "<i>Zahlenwert = ".$zahlenwert."</i>";
            echo "<br><br>";
        }
    }
    else
        echo "Fehler bei der Datenbankverbindung aufgetreten!";
?>

<p>
    <a href="index.php">Zurück zu den Zahlenrätseln von Ephesos!</a>
</p>

<?
    // Fußzeile inkludieren
    include ("footer.php");
?>

</body>
</html>
```

Abbildung 112 „admin.php“ (Teil 2)

Datei „admin2.php“

In dieser Datei werden, ähnlich wie in der „admin.php“-Datei, weitere Tests mit der Datenbank durchgeführt. Der HTML-Teil dieser Datei ist hier ebenfalls gleich wie bei den anderen Seiten, nur dass hier das Adminmenü („adminmenue.php“) eingebunden wird.

Zuerst wird die Datenbankverbindung inkludiert und aufgebaut. Im nächsten Schritt werden die ersten 50 Einträge der Tabelle „tbl_grnamen“ ausgelesen und in einer scrollbaren Box (definiert in der „styles.css“-Datei) angezeigt, siehe Abb. 113.

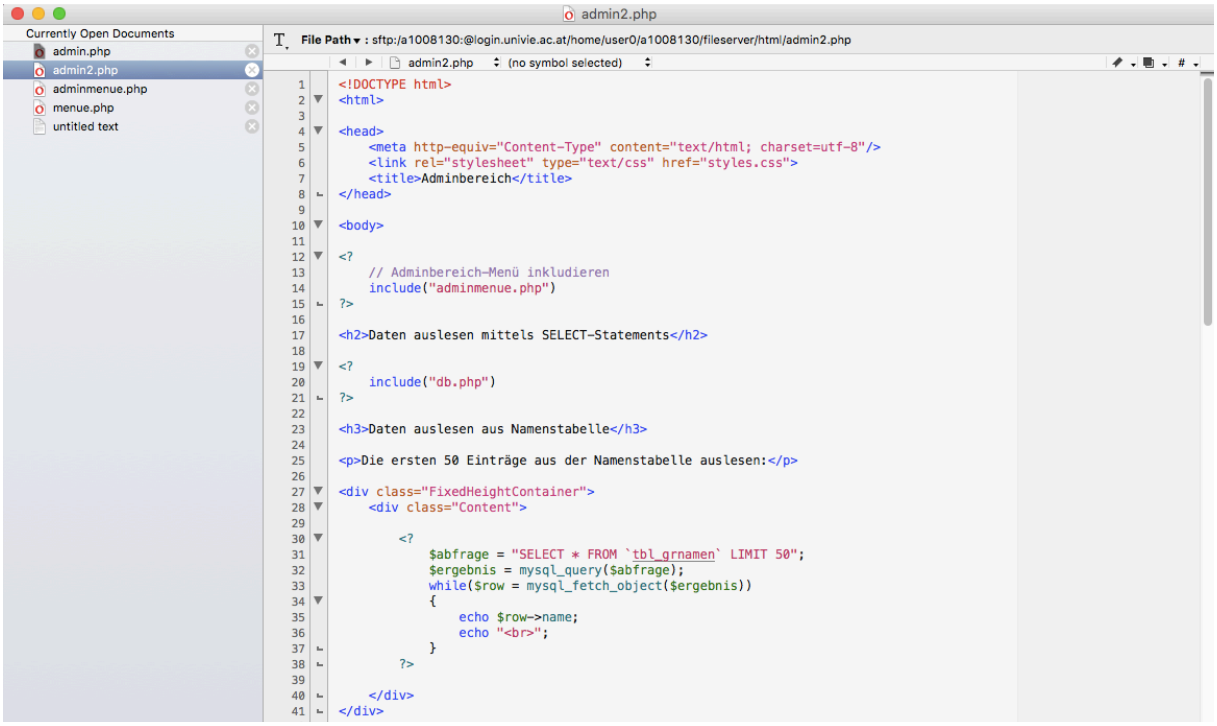


Abbildung 113 „admin2.php“ (Teil 1)

Dann wird die vollständige Entschlüsselungstabelle, aufsteigend nach den Zahlenwerten geordnet, ausgegeben (ebenfalls in einer scrollbaren Box).

Bei den nächsten `SELECT`-Abfragen werden zuerst alle Namen, in welchen der Großbuchstabe Alpha vorkommt, aus der Datenbank geholt und angezeigt, und anschließend alle Namen mit einem Eta-Großbuchstaben – die Abfragen werden allerdings beide Male limitiert auf 50 Datensätze. Am Ende werden die Daten freigegeben und die Datenbankverbindung geschlossen (siehe Abb. 114 und Abb. 115).

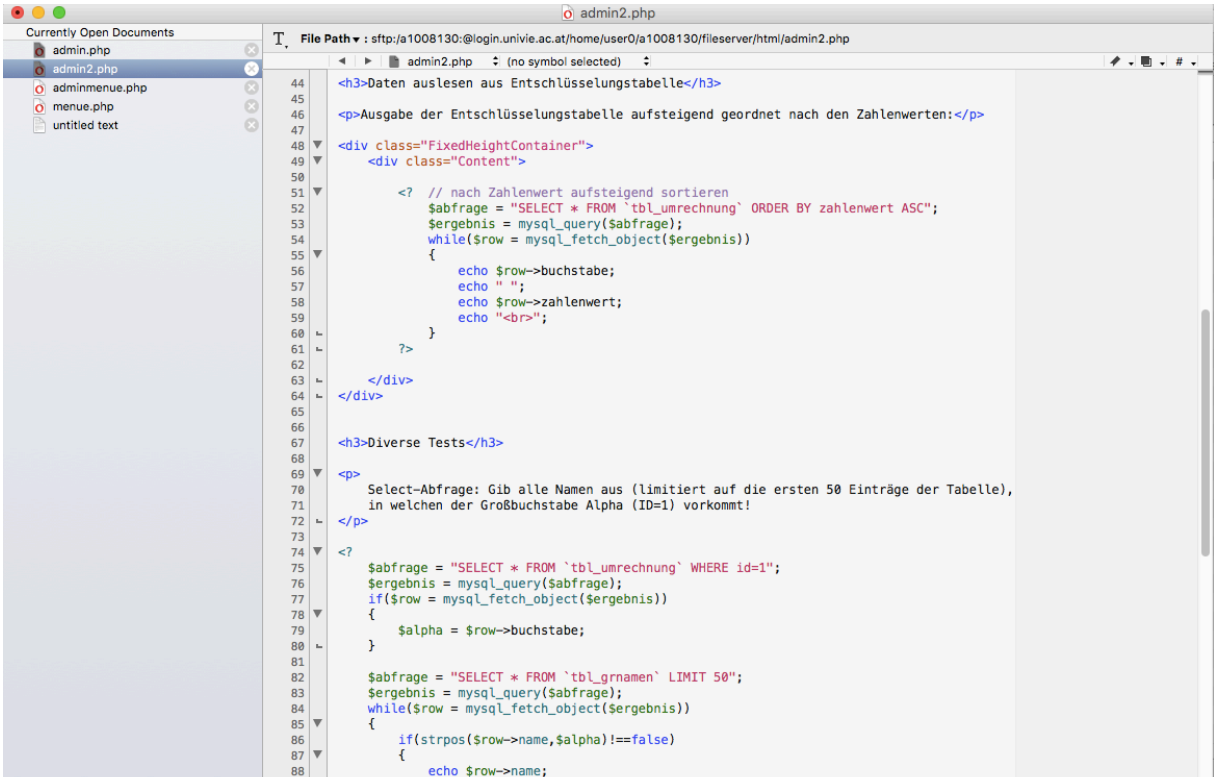


Abbildung 114 „admin2.php“ (Teil 2)

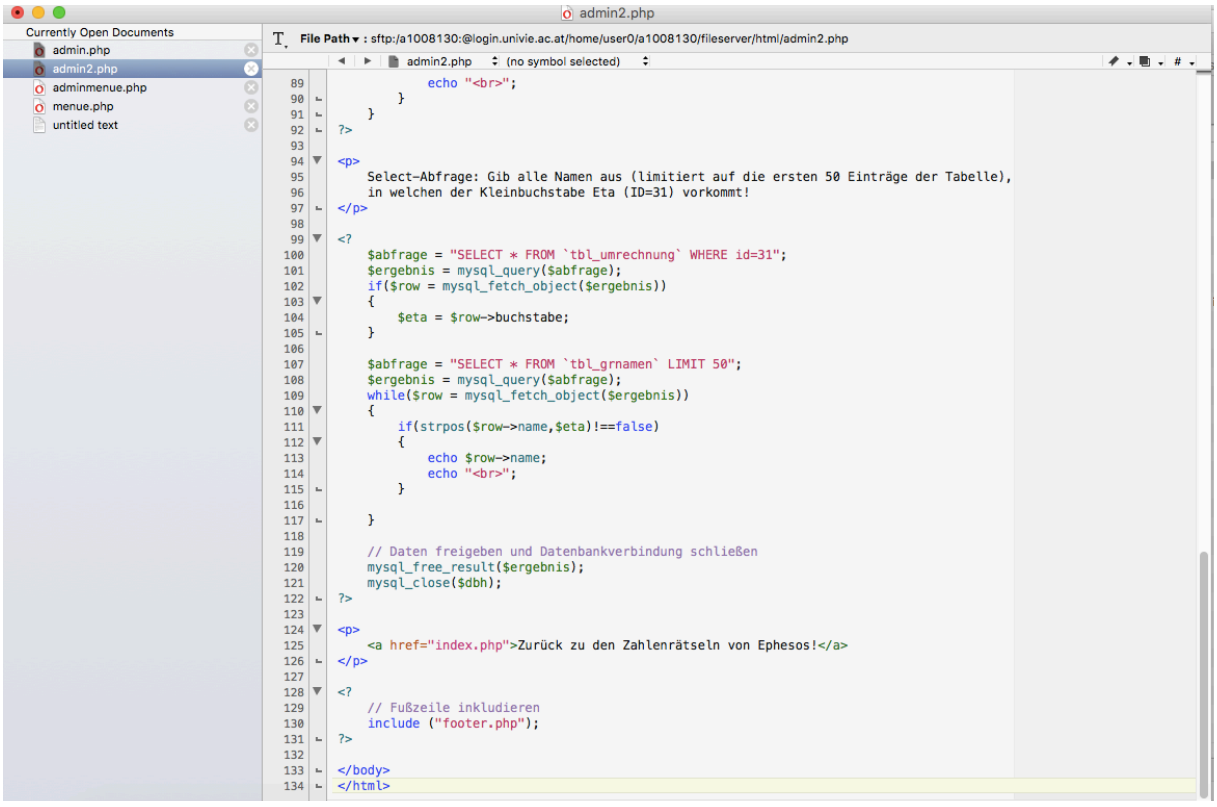


Abbildung 115 „admin2.php“ (Teil 3)

Datei „insert.php“

Diese Datei ist für das Berechnen und Einfügen der berechneten Zahlenwerte eines jeden griechischen Namens in die Tabelle „tbl_grnamen“ verantwortlich.

Zuerst wird die Verbindung zur Datenbank hergestellt, danach alle Datensätze aus der Umrechnungstabelle („tbl_umrechnung“) geholt und in ein Array gespeichert (Vorteil: bessere Laufzeit). Als Nächstes werden alle Namen aus der Datenbank geholt („tbl_grnamen“) und für jeden Buchstaben eines Namens wird der entsprechende Zahlenwert addiert. Auf diese Weise wird der Gesamtwert für jeden einzelnen Namen berechnet und der jeweilige Datensatz in der Tabelle upgedatet. Am Ende erfolgt die Ausgabe der Anzahl der erfolgreich upgedateten Datensätze (siehe Abb. 116 und Abb. 117).

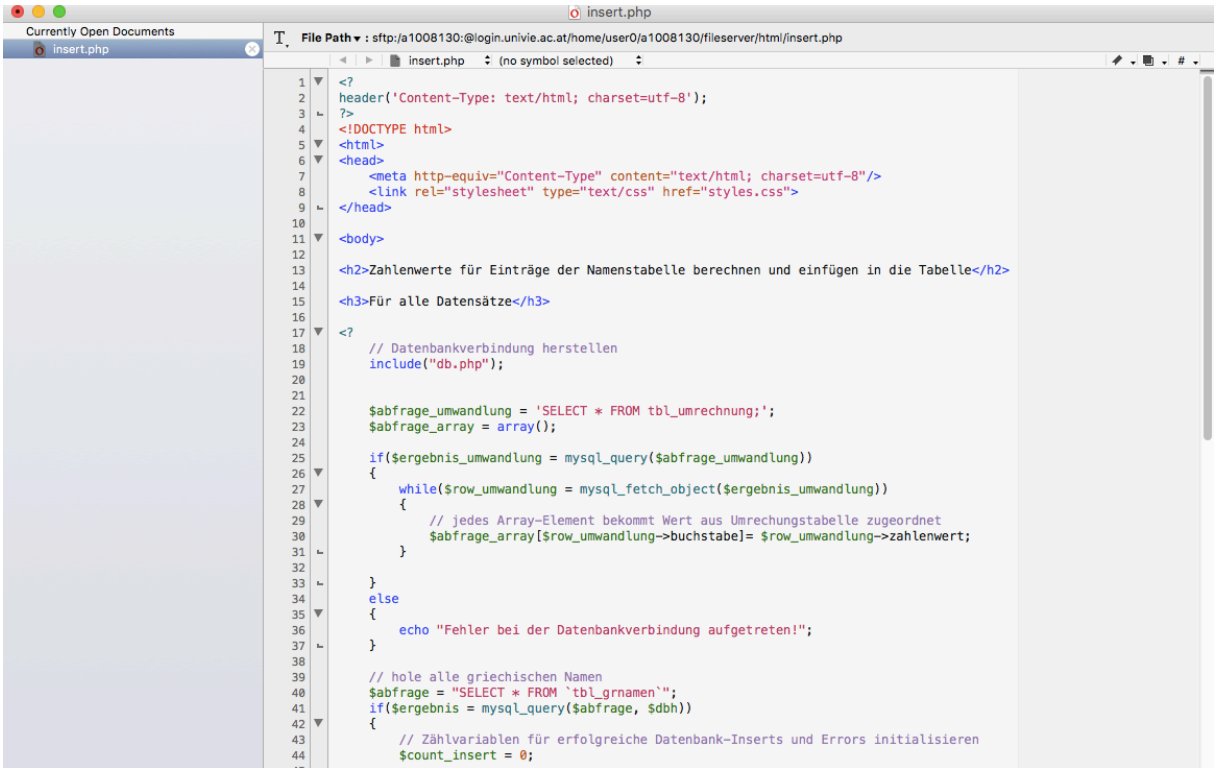


Abbildung 116 „insert.php“ (Teil 1)

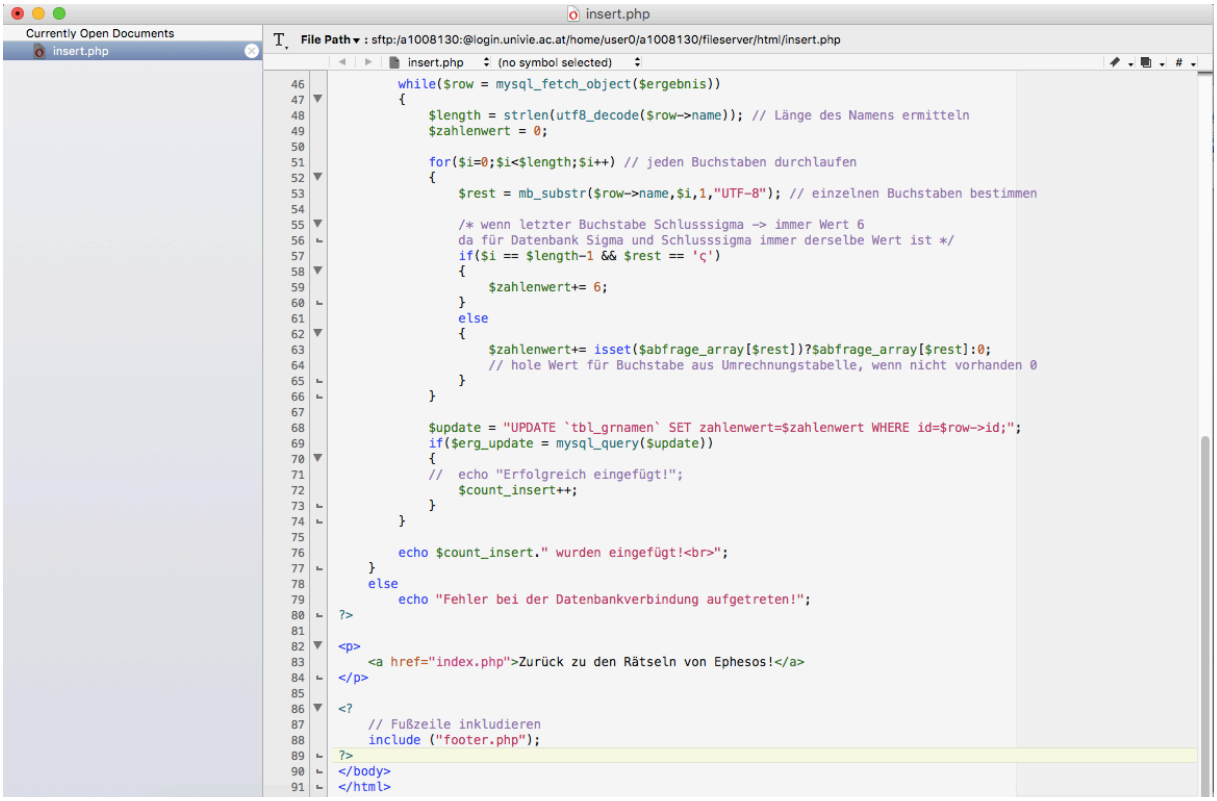


Abbildung 117 „insert.php“ (Teil 2)

Datei „adminmenue.php“

Diese Datei ist zur Einbettung in andere Dateien gedacht (vgl. „menue.php“). Sie gibt die Überschrift vom Adminbereich des Web Interfaces und die obere Menüstruktur aus und ist weiters für das Logout aus diesem zuständig. Sie verhält sich vom Aufbau her prinzipiell gleich die Datei „menue.php“, welche bereits erklärt wurde.

Abbildung 118 Code „adminmenue.php“ (Teil 1)

```
28  <h1>Adminbereich</h1>
29
30  <p align="right">
31      <a href="admin.php?logout">Logout</a>
32  </p>
33
34  <table width="100%">
35      <tr>
36          <?
37              foreach($menue as $entry => $datei)
38              // Menüeinträge einzeln durchgehen und anzeigen
39              {
40                  if ($datei == basename($_SERVER['PHP_SELF']))
41                  // gerade ausgewählten Menüeintrag kennzeichnen
42                  {
43                      echo "<td align=\"center\" bgcolor=\"#8B5A2B\" id=\"menue1\">";
44                      echo "$entry";
45                  }
46                  else
47                  {
48                      echo "<td align=\"center\" bgcolor=\"#CD8500\" id=\"menue2\">";
49                      echo "<a href=\"$datei\">$entry</a>";
50                      echo "</td>";
51                  }
52              }
53          ?>
54      </tr>
55  </table>
```

Abbildung 119 Code „adminmenue.php“ (Teil 2)

Dateien für klickbaren Keyboardersatz: „keyboard2.js“ und „keyboard2.js.css“

Der klickbare Keyboardersatz wurde auf folgender Webseite gefunden:

http://www.greywyvern.com/code/javascript/keyboard (aufgerufen am 1.5.2016)

Es handelt sich dabei um zwei einzubindende Dateien, wobei die „.js“-Datei für die Funktionalität zuständig ist und die „.css“-Datei für das Layout. Um den klickbaren Keyboardersatz an ein Textfeld zu hängen, wird die Klasse „keyboardInput“ zugewiesen.

4.5.3 Ablauf des Programms

In diesem Unterkapitel soll der Ablauf des Programms erläutert und zugleich veranschaulicht werden. Dieser „Rundgang“ läuft chronologisch ab und beginnt beim Login zur Startseite des Web Interfaces. Anschließend wird jeweils eine Zahl- und eine Namenseingabe gezeigt. Dann erfolgt eine Miteinbeziehung von diversen Ausnahmen – zuerst bei der Zahleingabe, danach bei der Namenseingabe. Ein Hinweis auf das Logout rundet die Veranschaulichung des Ablaufs schließlich ab.

Login

Gibt man im Browser die URL der Seite ein (http://homepage.univie.ac.at/a1008130), so gelangt man zur Login-Maske des Projekts, wie in Abb. 120 ersichtlich (das Login war allerdings beim Abschluss des Projekts nicht mehr relevant).

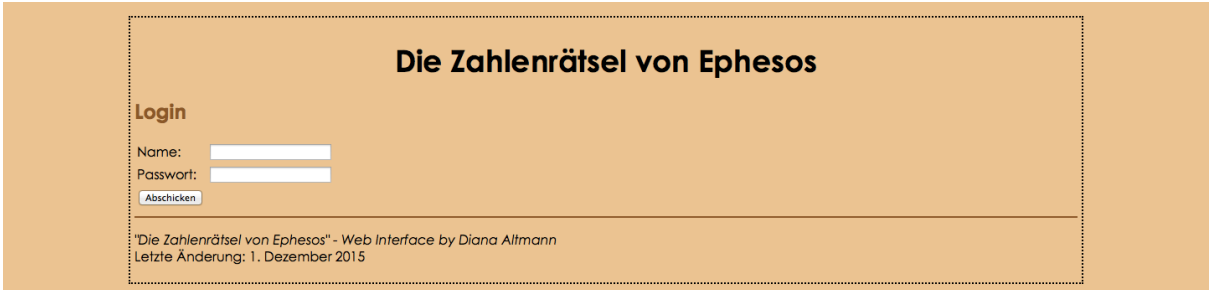


Abbildung 120 Login-Bildschirm

Nach Eingabe der richtigen Kombination aus Name und Passwort wird man dann zur Startseite des Web Interfaces weitergeleitet (siehe Abb. 121).

Abbildung 121 Startseite des Web Interfaces

Eingabe einer Zahl

Will man sich alle griechische Namen, die einer bestimmten Zahl zugeordnet sind, anzeigen lassen, so tippt man diese Zahl in das dafür vorgesehene Feld und klickt anschließend auf „Abschicken“ (siehe Abb. 122).

Abbildung 122 Eintragen der Zahl „1000“

Alle entsprechenden Namen werden in einer scrollbaren Box darunter angezeigt, wie in Abb. 123 ersichtlich:

Abbildung 123 Ergebnisse der Suche

Eingabe eines Namens

Will man den Zahlenwert eines bestimmten griechischen Namens berechnen lassen, so tippt man diesen Namen in das dafür vorgesehene Feld und klickt anschließend auf „Abschicken“. Das Eintippen wird durch den klickbaren Keyboardersatz erleichtert (Copy-Paste aus einem anderen Dokument ist natürlich ebenfalls möglich).

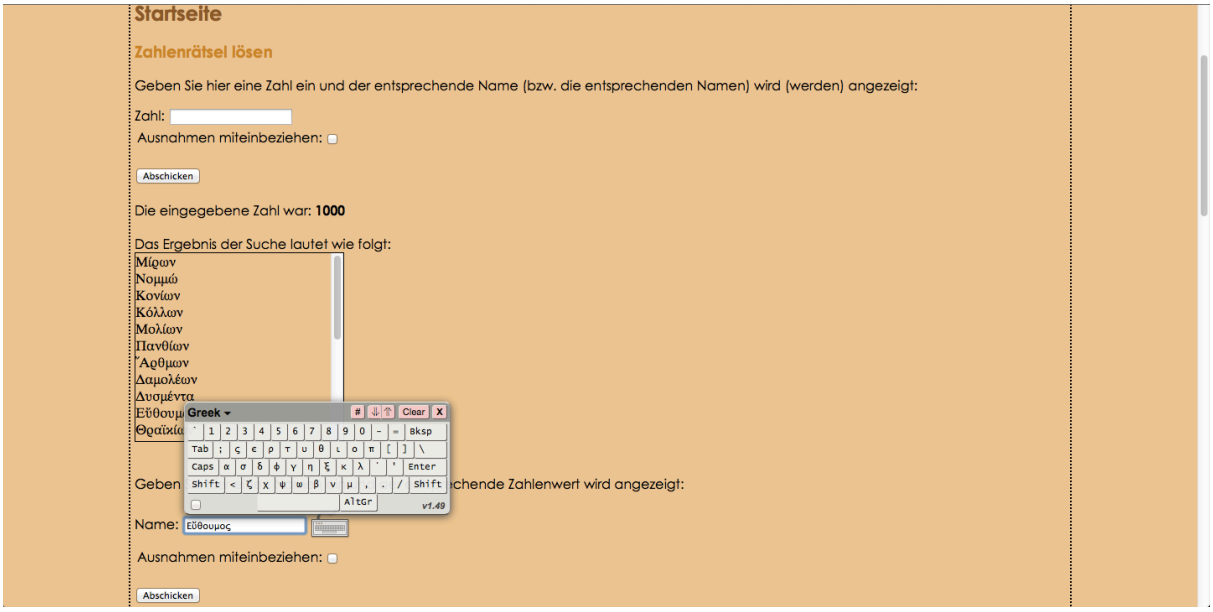


Abbildung 124 Einen griechischen Namen eingeben

Abbildung 125 Berechnung des Zahlenwerts für den eingegebenen Namen

Ausnahmen miteinbeziehen

Sowohl bei Zahl als auch bei Name ist es möglich, bestimmte (von Herrn ao. Univ.-Prof. Dr. Taeuber festgelegte) Ausnahmen miteinzubeziehen. Wird das Häkchen gesetzt, so öffnet sich ein Bereich mit allen auswählbaren Ausnahmen, welche beliebig miteinander kombiniert werden können (durch Setzen von Häkchen in Checkboxen), wie in Abb. 126 ersichtlich. Nun zu den zwei Anwendungsbereichen der Ausnahmen – Zahl und Name.

Zahl eingeben mit Ausnahmen

Abbildung 126 Eingabe der Zahl „100“ mit Ausnahmen (Teil 1)

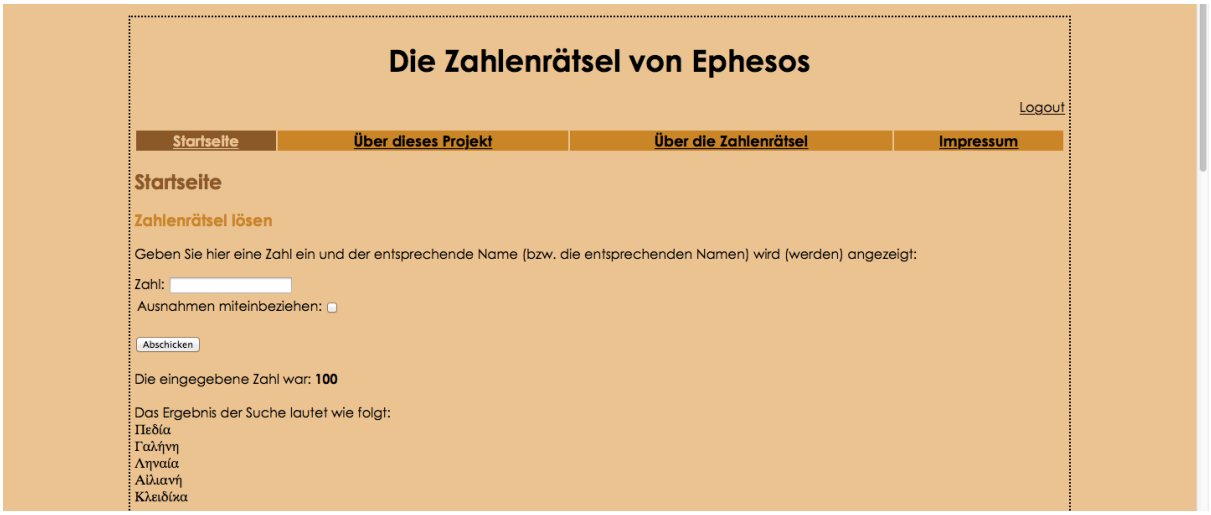


Abbildung 127 Eingabe der Zahl „100“ mit Ausnahmen (Teil 2)

Name eingeben mit Ausnahmen

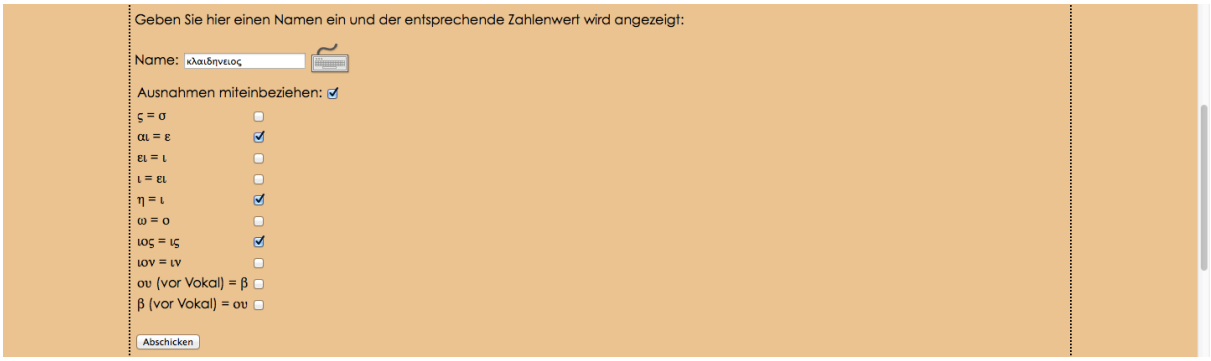


Abbildung 128 Eingabe eines Namens mit Ausnahmen (Teil 1)

Abbildung 129 Eingabe eines Namens mit Ausnahmen (Teil 2)

Logout

Klickt man rechts oben im Menü auf den „Logout“-Link, so gelangt man erneut zur Login-Maske des Web Interfaces. Das Ausloggen war erfolgreich.

4.6 Ergebnisse

Mit dem erstellten Programm zur Entschlüsselung der Zahlenrätsel von Ephesos konnten in der Forschung bereits Ergebnisse erzielt werden. So konnte von Herrn ao. Univ.-Prof. Dr. Taeuber eine überraschende Entdeckung gemacht werden, welche sich auf die ominöse Zahl

666 bezieht. Im Folgenden befindet sich nun der Artikel, der dazu im Medienportal der Universität Wien erschienen ist.[170]

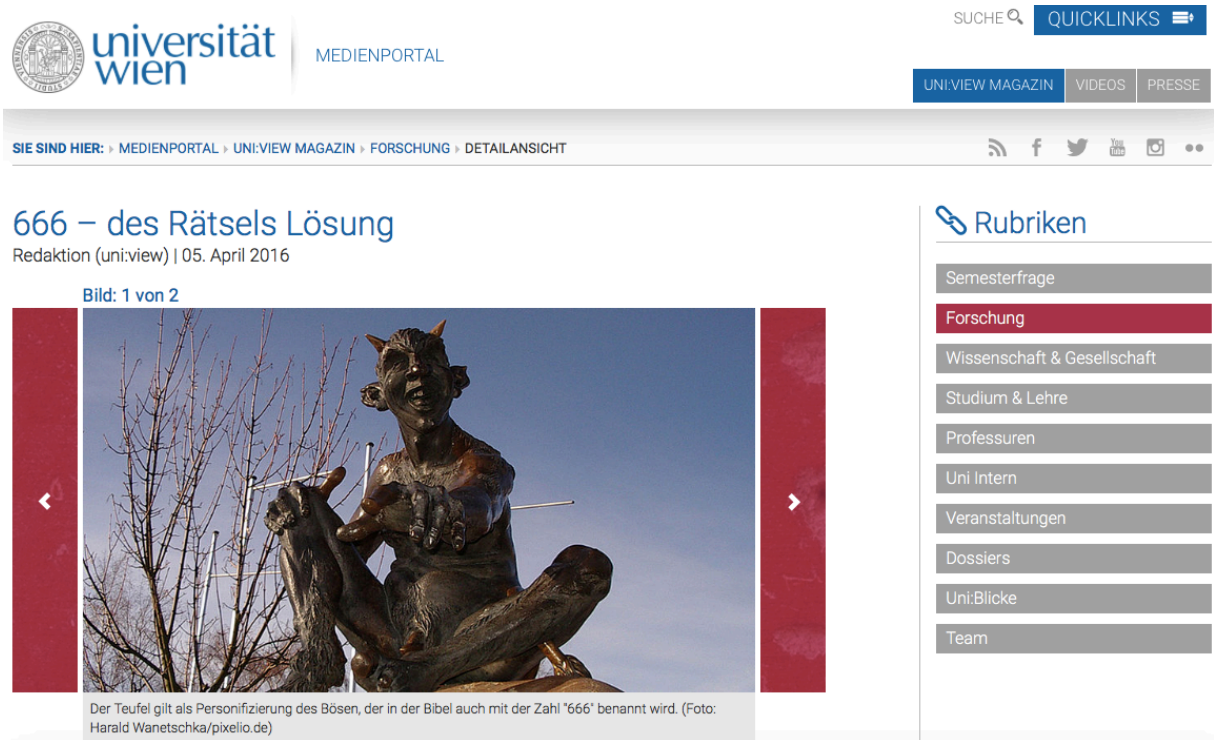


Abbildung 130 Artikel (Teil 1)

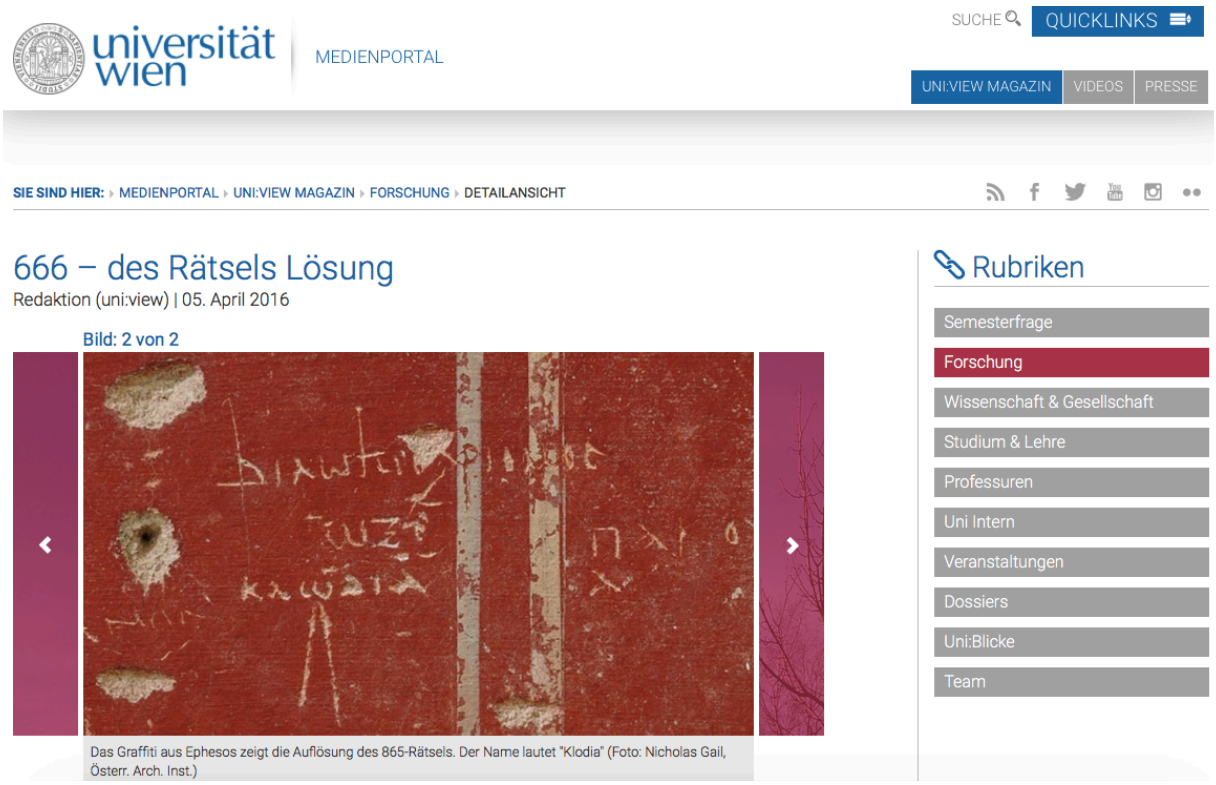


Abbildung 131 Artikel (Teil 2)

[170] http://medienportal.univie.ac.at/uniview/forschung/detailansicht/artikel/666-des-raetsels-loesung/, aufgerufen am 1.5.2016.

Lange Zeit rätselten WissenschafterInnen, wer sich hinter dem "Untier" der biblischen geheimen Offenbarung, das mit 666 umschrieben wird, verbirgt. Hans Taeuber vom Institut für Alte Geschichte kam nun dem größten Rätsel der Bibel auf die Spur: Gemeint sei Kaiser Trajan.

"Ich liebe die, deren Zahl 865 ist". So die Übersetzung eines der Wandgraffiti, die im "Hanghaus 2" bei Ausgrabungen in Ephesos zum Vorschein kamen. Diese sogenannten "isopsephischen Rätsel" beruhen auf dem Prinzip, dass jeder Buchstabe des griechischen Alphabets zugleich einen Zahlenwert hat; so steht Alpha für eins, Beta für zwei usw. Durch Addition der Zahlwerte der Buchstaben, aus denen ein Name besteht, ergibt sich eine Summe, aus der der Eingeweihte den ursprünglichen Namen erschließen konnte. Während für die antiken TeilnehmerInnen an diesem "Partyspiel" die Zahl der in Frage kommenden Personen ziemlich überschaubar war, ist es für heutige ForscherInnen ohne technische Hilfsmittel praktisch unmöglich, die Lösung solcher Rätsel zu ermitteln.

Der Name hinter der Zahl

Diana Altmann, Studentin der Informatik, hat unter Anleitung von Hans Taeuber vom Institut für Alte Geschichte in ihrer Diplomarbeit ein Programm entwickelt, mit dessen Hilfe die Zahlenwerte aller in Kleinasien nachgewiesenen Personennamen berechnet werden können. Die entsprechende Datenbasis in elektronischer Form wurde von Richard Catling, dem Autor des "Lexicon of Greek Personal Names", zur Verfügung gestellt. "Damit ist es uns nun möglich, zu jeder 'Rätselzahl' eine Liste antiker Personennamen zu erstellen, auf die die betreffende Quersumme zutrifft. Für die Berechnung kann eine beliebige Kombination orthographischer Varianten berücksichtigt werden", erklärt Taeuber.

Abbildung 132 Artikel (Teil 3)

"... und seine Zahl ist 666"

Mit Hilfe dieses Programms gelang es endlich, einem der größten Rätsel der Bibel auf die Spur zu kommen: In Kapitel 13 der Geheimen Offenbarung wird ein "Untier" (therion) beschrieben, dem der "Drache" (Satan) große Macht verliehen hat und dessen Bild alle Menschen anbeten müssen. Und weiter heißt es: "Hier ist die Weisheit. Wer Verstand hat, berechne die Zahl des Tieres, denn es ist eines Menschen Zahl, und seine Zahl ist 666."

Über die Rätsel von Ephesos: Die Buchstaben des griechischen Alphabets hatten neben ihrem Laut- auch einen Zahlenwert. Das System war in drei Neunergruppen gegliedert, je eine Gruppe repräsentierte die Einer, Zehner und Hunderter. Es begann mit Alpha = 1, Beta = 2 usw. bis Theta = 9; dann setzten die Zehner mit Iota = 10, Kappa = 20 … fort, und schließlich folgten die Hunderter mit Rho = 100, Sigma = 200 etc. Da das klassische griechische Alphabet nur 24 Buchstaben umfasste, hat das Zahlsystem drei damals in der Schrift bereits ungebräuchliche Buchstaben weiterverwendet: Stigma (ϛ) = 6, Koppa (Ϙ) = 90 und Sampi (ϡ) = 900. Ab 1.000 wurde wieder mit Alpha begonnen, allerdings mit diakritischem Zeichen (im Druck als ‚α wiedergegeben), um es vom einfachen α' = 1 zu unterscheiden.

"Ganz offensichtlich handelt es sich dabei um die gleiche Art von Rätseln, wie sie auf den etwa zur selben Zeit wie die Offenbarung entstandenen Wandmalereien zu lesen waren und die dem im westlichen Kleinasien ansässigen Autor vertraut sein mussten", sagt der Historiker: "Der Kontext legt nahe, dass es sich um einen römischen Kaiser handeln dürfte, und tatsächlich wurde das 'Untier' von den ForscherInnen über die Jahrhunderte hinweg immer wieder mit Nero, Domitian oder zuletzt auch Hadrian identifiziert." Die Eingabe der Zahl "666" in das Programm führt jedoch zu einem ganz anderen Ergebnis: nämlich "Ulpius", der Familienname des Kaisers Marcus Ulpius Traianus, der von 98-117 n. Chr. regierte.

Abbildung 133 Artikel (Teil 4)

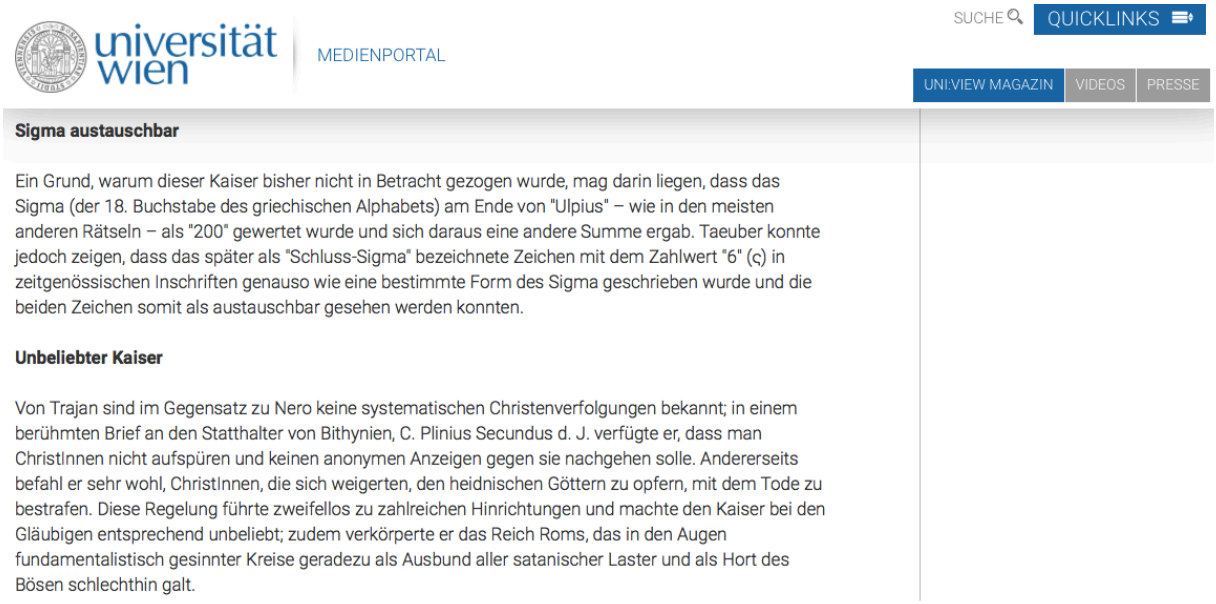


Abbildung 134 Artikel (Teil 5)

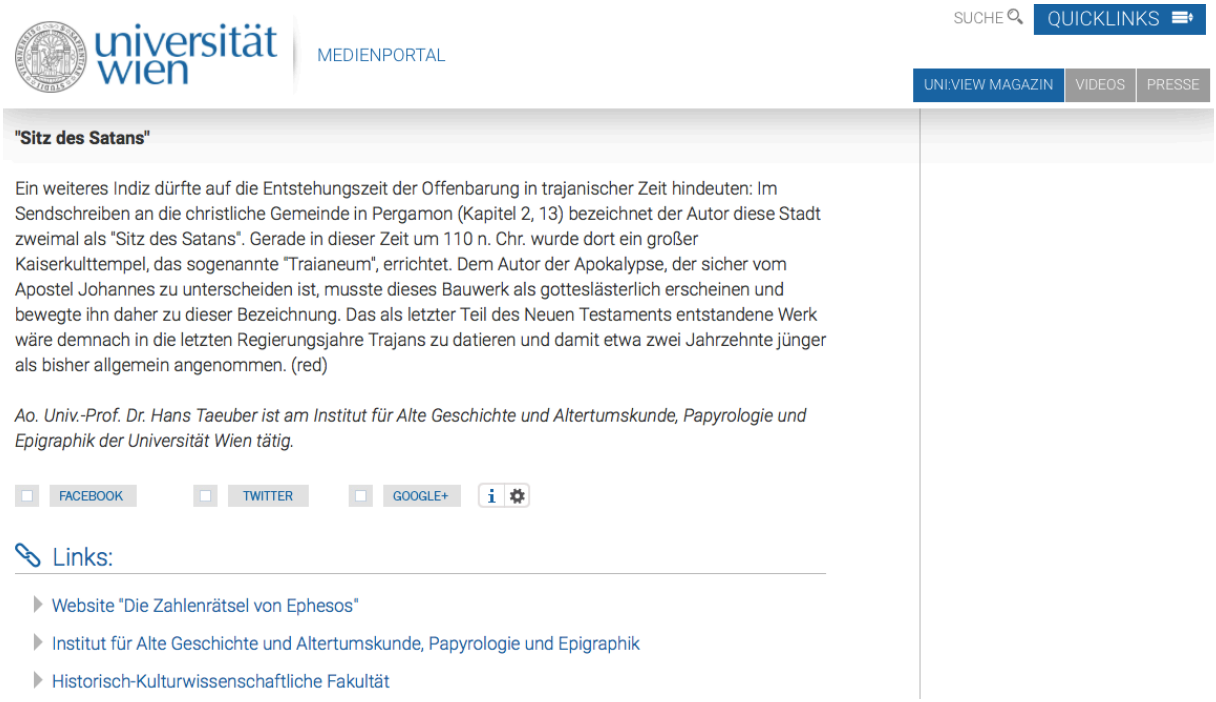


Abbildung 135 Artikel (Teil 6)

Doch nicht nur vonseiten der Universität Wien wurde dieses Thema aufgegriffen, sondern auch diverse weitere Zeitungen und Webportale publizierten diese Ergebnisse in eigenen Artikeln. Es folgte regelrecht ein Medienecho, wie an folgender Liste (welche aber nur eine Auswahl darstellt) ersichtlich:

- http://www.science.apa.at/site/kultur_und_gesellschaft/detail.html?key=SCI_20160411_SCI39351351629199056 (aufgerufen am 1.5.2016)
- http://www.science.apa.at/site/kultur_und_gesellschaft/detail.html?key=SCI_20160411_SCI39431352629199024 (aufgerufen am 1.5.2016)

- http://diepresse.com/home/science/4965190/666-die-Zahl-des-Kaisers-Trajan (aufgerufen am 1.5.2016)
- http://derstandard.at/2000034621927/Forscher-wollen-das-Untier-hinter-der-Zahl-666-identifiziert-haben (aufgerufen am 1.5.2016)
- http://sciencev2.orf.at/stories/1769308/ (aufgerufen am 1.5.2016)
- http://www.wienerzeitung.at/themen_channel/wissen/geschichte/811938_Das-Zahlenraetsel-von-Ephesos.html (aufgerufen am 1.5.2016)
- http://www.austria.com/kaiser-trajan-soll-untier-666-in-der-biblischen-offenbarung-sein/4686148 (aufgerufen am 1.5.2016)
- http://www.vienna.at/kaiser-trajan-soll-untier-666-in-der-biblischen-offenbarung-sein/4686148 (aufgerufen am 1.5.2016)
- http://www.vol.at/kaiser-trajan-soll-untier-666-in-der-biblischen-offenbarung-sein/4686148 (aufgerufen am 1.5.2016)
- http://www.oe24.at/welt/Bibel-Geheimnis-um-666-geloest/231391107 (aufgerufen am 1.5.2016)
- http://www.nachrichten.at/nachrichten/chronik/Kaiser-Trajan-soll-Untier-666-in-biblischer-Offenbarung-sein;art58,2201675 (aufgerufen am 1.5.2016)
- http://www.scinexx.de/wissen-aktuell-20060-2016-04-12.html (aufgerufen am 1.5.2016)
- http://www.stol.it/Artikel/Panorama-im-Ueberblick/Panorama/Kaiser-Trajan-soll-Untier-666-in-der-biblischen-Offenbarung-sein (aufgerufen am 1.5.2016)
- http://motherboard.vice.com/de/read/biblisches-zahlenraetsel-geloest-forscher-wissen-wer-die-bestie-hinter-666-ist-antichrist-wien-777 (aufgerufen am 1.5.2016)
- http://www.oe-journal.at/index_up.htm?http://www.oe-journal.at/Aktuelles/!2016/0416/W2/51204uniWien.htm (aufgerufen am 1.5.2016)
- http://dradiowissen.de/nachrichten/antike-software-soll-alte-liebschaften-entschluesseln (aufgerufen am 1.5.2016)
- Facebook:
    - https://www.facebook.com/122737471962/posts/10154019404666963 (aufgerufen am 1.5.2016)
    - https://www.facebook.com/53969266331/posts/10153444510466332 (aufgerufen am 1.5.2016)
- Twitter:
    - https://twitter.com/univienna/status/719467252781817858 (aufgerufen am 1.5.2016)
    - https://twitter.com/univienna/statuses/719467252781817858 (aufgerufen am 1.5.2016)

5. Unterrichtsszenarien

In diesem Kapitel befinden sich die vier ausgearbeiteten Unterrichtsszenarien mit folgenden Titeln: „Grundlagen von Programmiersprachen anhand von PHP und dessen Einbindung in Webstrukturen“, „Fortgeschrittene Konzepte von Programmiersprachen anhand von PHP“, „Datenbankgrundlagen und Einbindung in PHP-Code“ und „Kryptographie: Theorie und Praxis“.

Wichtige Informationen: Die vorliegenden Szenarien bauen jeweils aufeinander auf und sollten deswegen in dieser Reihenfolge verwendet werden. Es ist aber auch möglich, einzelne Aspekte herauszunehmen und einzeln zu behandeln oder individuell Schwerpunkte zu setzen.

Aufbau der Szenarien: Die Szenarien enthalten jeweils allgemeine Informationen (Thema, Lehrplanbezug, angestrebte Stundenziele), eine Planungsmatrix (über den konkreten Ablauf), Theorieverweise (auf die jeweiligen Kapitel) und ausgewählte Code-Beispiele (als Screenshots). Die vollständigen Code-Dateien befinden sich auf beiliegender CD-ROM.

Planung: Wichtig bezüglich der Planung der einzelnen Unterrichtsszenarien war, dass sich ein roter Faden durch alle Szenarien zieht und auch, dass die Stunden jeweils in mehrere Phasen unterteilt werden, welche stets aufeinander aufbauen. Vier Phasen werden unterschieden und folgen jeweils aufeinander: Hinführungsphase, Erarbeitungsphase, Vertiefungsphase und Festigungsphase.

Kompetenzorientierter Unterricht: „Kompetenzorientierung bedeutet einen Perspektivenwechsel weg von einer Orientierung auf reinen Wissenserwerb hin zu einer intelligenten Anwendung von Wissen.“ [171] Man unterscheidet hierbei drei Anforderungsbereiche: Reproduktionsleistung, Transferleistung sowie Reflexion und Problemlösung. Es wurde versucht, diese drei Bereiche jeweils in die Planung miteinzubeziehen.

[171] URL: http://diepresse.com/home/politik/innenpolitik/forumbildung/759303/Von-der-Reproduktion-zur-Reflexion, aufgerufen am 21.5.2016.

5.1 Szenario 1: Grundlagen von Programmiersprachen anhand von PHP und dessen Einbindung in Webstrukturen

5.1.1 Allgemeines

Thema: HTML, CSS, PHP

Schulform: AHS Oberstufe

Lehrplanbezug:

Lehrstoff Wahlpflichtfach Informatik – 6. bis 8. Klasse[172]*:*

- Grundprinzipien der Informationsverarbeitung
- Konzepte von Betriebssystemen
- Aufbau und Funktionsweise von Netzwerken
- Datenbanken
- Lern- und Arbeitsorganisation
- Konzepte von Programmiersprachen
- künstliche Intelligenz
- Erweiterung der theoretischen und technischen Grundlagen der Informatik – grundlegende Algorithmen und Datenstrukturen
- Informatik, Gesellschaft und Arbeitswelt
- Rechtsfragen

Mit diesem Szenario wird Bezug genommen auf folgende zwei Aspekte des Lehrplans: „Konzepte von Programmiersprachen“ und „Erweiterung der theoretischen und technischen Grundlagen der Informatik – grundlegende Algorithmen und Datenstrukturen“.

Angestrebte Stundenziele:

Wissenserweiterung und Kompetenzerwerb: Die Schülerinnen und Schüler sollen vielseitiges Wissen und Kompetenzen zum Thema Web Interfaces erworben haben und den Unterschied zwischen einer Auszeichnungssprache (wie HTML) und Programmiersprache (wie z.B. PHP) kennen und erklären können und mit beiden umgehen bzw. sie anwenden können.

Sozialverhalten: Die Schülerinnen und Schüler sollen lernen, Ergebnisse vor der Klasse zu präsentieren, weiters sollen sie aufmerksam zuhören können, wenn Klassenkolleginnen und -kollegen sprechen und auch konstruktives Feedback geben und annehmen können.

[172] URL: https://www.bmbf.gv.at/schulen/unterricht/lp/lp_neu_ahs_21_11876.pdf?4dzgm2, aufgerufen am 2.12.2015.

5.1.2 Planungsmatrix

Ablauf (für eine Doppelstunde – 2 x 50 min):

Zeit	Inhalt und Phase	Sozialform	Material
2-3 Min.	warten, bis alle Schülerinnen und Schüler in der Klasse angekommen sind; Begrüßung	L-S-Gespräch(e)	–
20 Min.	*Hinführungsphase:* „Die Zahlenrätsel von Ephesos"-Web Interface den Lernenden zeigen, etwas über die Zahlenrätsel erzählen, Verschlüsselung der Rätsel erklären und zeigen → Interesse wecken zu den Themen HTML, CSS, PHP, Datenbanken, Verschlüsselung; etwaiges Vorwissen aktivieren und zur Sprache kommen lassen, Zahl- und Namenseingabe ausprobieren lassen	L-S-Gespräch	Beamer, PC
35 Min.	*Erarbeitungsphase*: Webserver erklären und starten, Grundgerüst für Homepage erstellen („index.php" und „uebermich.php"), Menü erstellen („menue.php"), CSS Datei („styles.css") anlegen → siehe Code-Beispiele	gemeinsames Erarbeiten L+S	Beamer, PCs
20 Min.	*Vertiefungsphase*: Individualisieren der Homepages → weitere Seiten anlegen und befüllen, Menü erweitern, CSS individualisieren	Einzelarbeit	Beamer, PCs
20 Min.	*Festigungsphase*: Schülerinnen und Schüler, die bereits fertig sind, dürfen ihre Homepages kurz präsentieren per Beamer; Feedback von L an S und von S an S	Präsentationen	Beamer, PC
2-3 Min.	Aufgabe bis zur nächsten Einheit für Mitarbeitsplus: wer noch nicht fertig geworden ist → Homepages zu Hause weiter gestalten bzw. befüllen (mind. 3 Seiten + Impressum); Ausblick auf kommende Einheit geben (fortgeschrittene Konzepte von Programmiersprachen); Verabschiedung	L-Gespräch	–

Tabelle 4 Planungsmatrix – Szenario 1

5.1.3 Ausarbeitung

Theorie

Die notwendige Theorie für dieses Unterrichtsszenario kann dem Kapitel „3.1 PHP" entnommen werden – vor allem sind „3.1.3 Strukturen einer PHP-Seite", „3.1.4 Variablen", „3.1.5 Kommentare", „3.1.7 Kontrollstrukturen" und „3.1.9 Einbinden externer Dateien" relevant. Weiters wichtig in Bezug auf die technischen Grundlagen ist auch das Kapitel „3.3 Webserver". Der geschichtliche Hintergrund findet sich im Kapitel 6.

Beschreibung der Unterrichtsphasen

Hinführungsphase:

Den Lernenden wird von der Lehrperson das Web Interface „Die Zahlenrätsel von Ephesos“ gezeigt und Diverses dazu erläutert, z.B. der Aufbau, die Menüstruktur, die Felder zum Zahl bzw. Name eingeben, die Datenbank dahinter, usw. Des Weiteren wird den Schülerinnen und Schülern etwas über die Zahlenrätsel erzählt und die Art der Verschlüsselung dieser Rätsel erklärt und gezeigt.

Diese Phase ist dazu da, um Interesse an den Themen HTML, CSS, PHP, Datenbanken und Verschlüsselung zu wecken. Weiters soll diese Einführung etwaiges Vorwissen zu diesen soeben angesprochenen Themen aktivieren und auch zur Sprache kommen lassen (die Schülerinnen und Schüler sollen beispielsweise gefragt werden, ob sie selbst schon einmal eine Homepage mit HTML und CSS erstellt haben). Auch die Komponente des Selbstausprobierens und Selbstaktivwerdens kommt in dieser Phase nicht zu kurz, denn so ist es den Lernenden erlaubt, sowohl die Zahleingabe als auch die Namenseingabe/ Zahlenberechnung selbst auszuprobieren mit Hilfe des implementierten Keyboardersatzes.

Erarbeitungsphase:

In dieser Phase wird gemeinsam mit den Schülerinnen und Schülern das Grundgerüst für das Web Interface erstellt, dazu muss jedoch zuerst das Thema „Webserver“ angesprochen und dieser anschließend gestartet werden. Dann werden gemeinsam die „index.php“-, die „menue.php“- und die „uebermich.php“-Datei erstellt. Des Weiteren wird eine CSS-Datei angelegt („styles.css“). Es werden grundsätzliche Theorieaspekte erklärt, die im Code enthalten sind.

Vertiefungsphase:

Als nächster Schritt – und zugleich als Vertiefung des soeben gelernten – dürfen/sollen die Lernenden ihr Web Interface ganz nach ihrem Belieben individualisieren. Das bedeutet, dass sowohl weitere Seiten angelegt und befüllt werden sollen mit diversen Inhalten (Texten, Bildern, Videos, Links, usw.). Diese neuen Seiten sollen natürlich auch in das Menü integriert werden. Auch die CSS-Datei darf/soll individualisiert werden.

Festigungsphase:

In dieser Phase dürfen Lernende, die bereits fertig sind, ihre Web Interfaces den Klassenkolleginnen und -kollegen kurz per Beamer präsentieren. Vorteile dieser Präsentationen sind einerseits, dass sich die Klassenkameradinnen und -kameraden dadurch zuhören und konstruktives Feedback geben lernen und sich noch Ideen holen können;

andererseits natürlich auch das Präsentieren an sich und das Feedback annehmen können. Das Feedback von der Lehrperson an die Präsentierende bzw. den Präsentierenden spielt natürlich auch eine große Rolle – so soll neben Verbesserungsvorschlägen natürlich auch das Loben von dem, was gut gelungen ist, nicht zu kurz kommen.

5.1.4 Code-Beispiele

Für dieses Unterrichtsszenario werden vier Dateien benötigt – „menue.php“, „index.php“, „uebermich.php“ und „styles.css“, welche beiliegender CD-ROM enthalten sind (Ordner „Szenario 1“).

Datei „menue.php“:

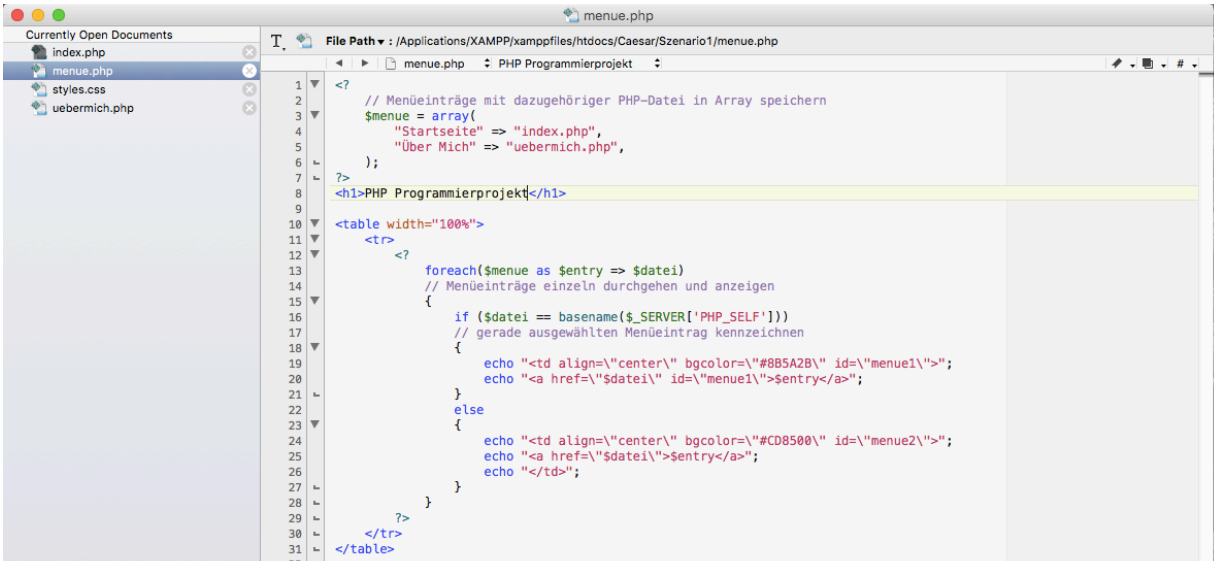


Abbildung 136 Szenario 1 – „menue.php“

Datei „index.php“:

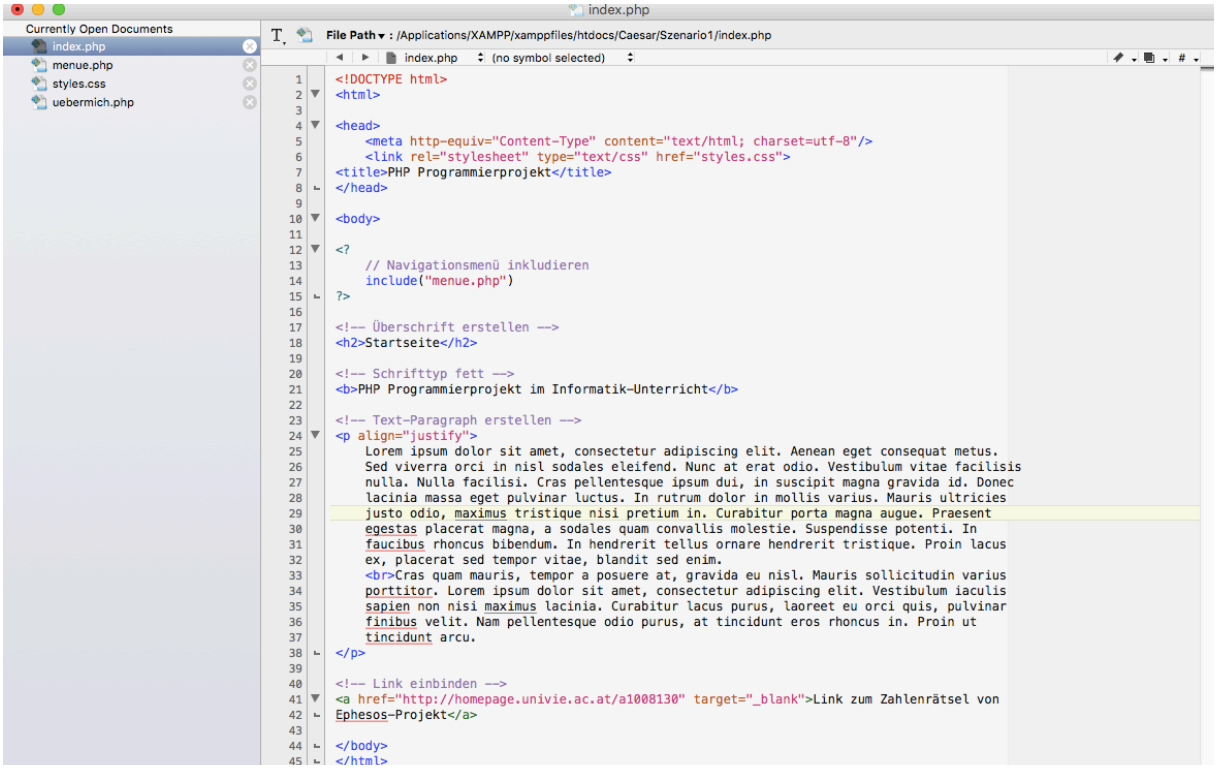


Abbildung 137 Szenario 1 – „index.php“

Datei „uebermich.php“:

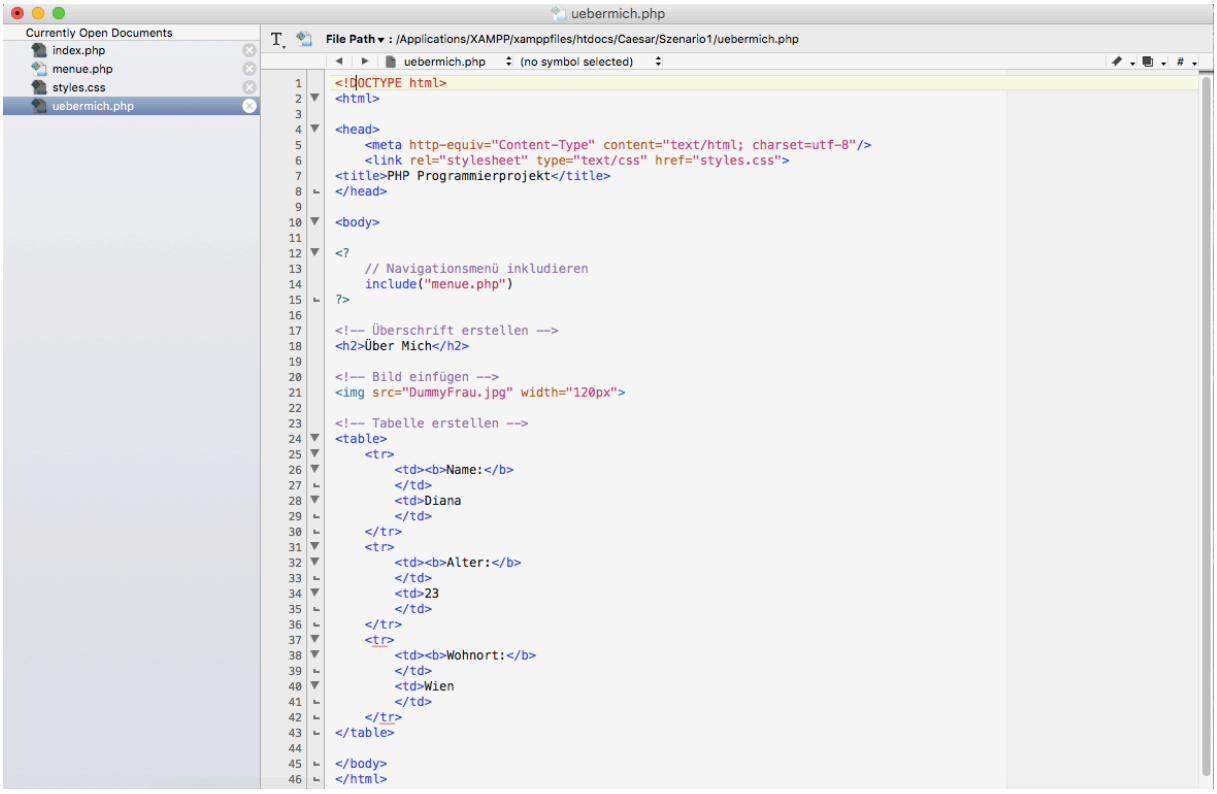


Abbildung 138 Szenario 1 – „uebermich.php“

Datei „styles.css“:

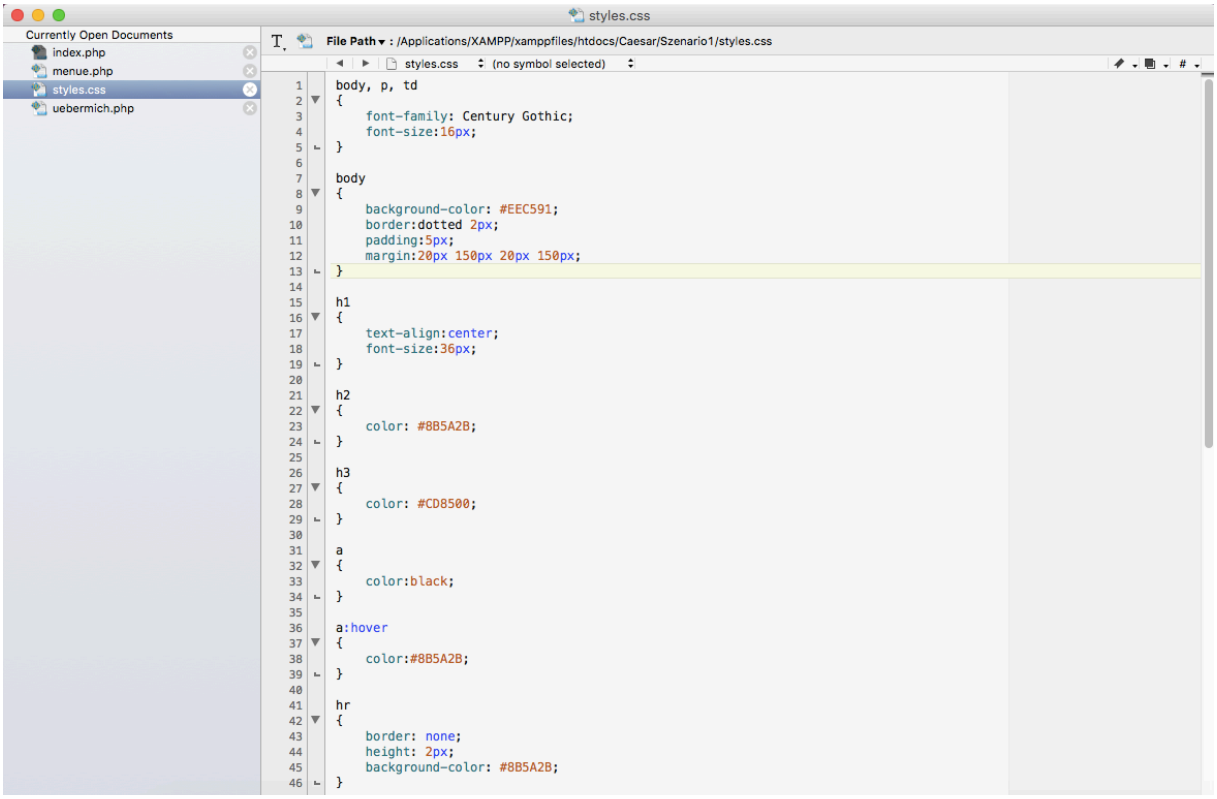


Abbildung 139 Szenario 1 – „styles.css“ (Teil 1)

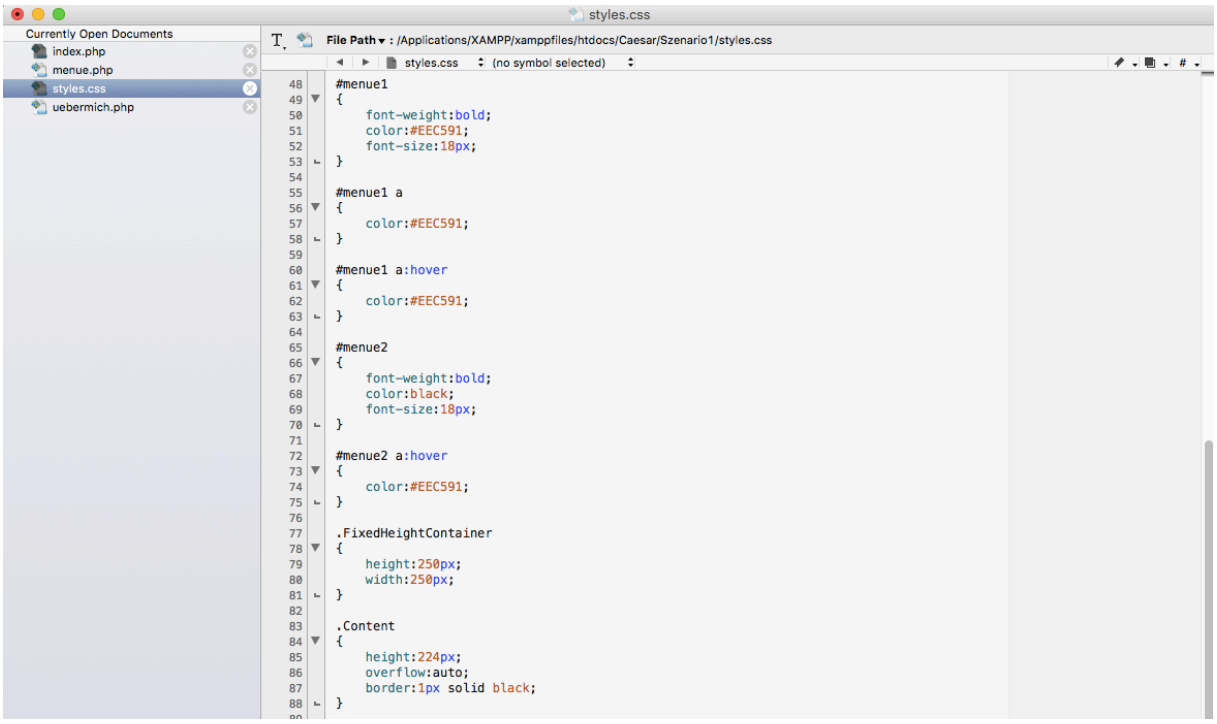


Abbildung 140 Szenario 1 – „styles.css“ (Teil 2)

5.2 Szenario 2: Fortgeschrittene Konzepte von Programmiersprachen anhand von PHP

5.2.1 Allgemeines

Thema: HTML, CSS, PHP

Schulform: AHS Oberstufe

Lehrplanbezug:

Lehrstoff Wahlpflichtfach Informatik – 6. bis 8. Klasse[173]*:*

- Grundprinzipien der Informationsverarbeitung
- Konzepte von Betriebssystemen
- Aufbau und Funktionsweise von Netzwerken
- Datenbanken
- Lern- und Arbeitsorganisation
- Konzepte von Programmiersprachen
- künstliche Intelligenz
- Erweiterung der theoretischen und technischen Grundlagen der Informatik – grundlegende Algorithmen und Datenstrukturen
- Informatik, Gesellschaft und Arbeitswelt
- Rechtsfragen

Mit diesem Szenario wird erneut Bezug genommen auf die folgenden zwei Aspekte des Lehrplans: „Konzepte von Programmiersprachen“ und „Erweiterung der theoretischen und technischen Grundlagen der Informatik – grundlegende Algorithmen und Datenstrukturen“.

Angestrebte Stundenziele:

Wissenserweiterung und Kompetenzerwerb: Die Schülerinnen und Schüler sollen vielseitiges Wissen und Kompetenzen zum Thema fortgeschrittene Konzepte von Programmiersprachen anhand von PHP (Session, Funktionen, Kontrollstrukturen, Requestmethoden GET und POST, ...) erworben haben und grundlegende Algorithmen anwenden können.

Sozialverhalten: Die Schülerinnen und Schüler sollen lernen, Ergebnisse vor der Klasse zu präsentieren. Des Weiteren sollen sie aufmerksam zuhören können, wenn Klassenkolleginnen und -kollegen sprechen und auch konstruktives Feedback geben und annehmen können.

[173] URL: https://www.bmbf.gv.at/schulen/unterricht/lp/lp_neu_ahs_21_11876.pdf?4dzgm2, aufgerufen am 2.12.2015.

5.2.2 Planungsmatrix

Ablauf (für eine Doppelstunde – 2 x 50 min):

Zeit	Inhalt und Phase	Sozialform	Material
2-3 Min.	warten, bis alle Schülerinnen und Schüler in der Klasse angekommen sind; Begrüßung	L-S-Gespräch(e)	–
20 Min.	es wird eine Wiederholung mit den Lernenden durchgeführt über die wichtigsten Konzepte der letzten Einheit; weiters werden Fragen bezüglich der Hausübung beantwortet bzw. gemeinsam besprochen und auf freiwilliger Basis kurze Präsentationen der Projekte durchgeführt	L-S-Gespräch	Beamer, PC
20 Min.	*Hinführungsphase:* Theorie zu fortgeschrittenen Aspekten → Funktionen; Requestmethoden (GET und SET) bei Formularen; Kontrollstrukturen; erklären, warum eine Session notwendig ist (für Login → Schutz vor Fremdzugriff), wichtige Variablen in Bezug auf Session, Überblick über bessere Programmierung geben (Funktionen → Vorteile: Übersicht, Wiederverwendbarkeit von Code)	L-Vortrag	Beamer, ev. Handouts
20 Min.	*Erarbeitungsphase*: diverse Funktionen inklusive Kontrollstrukturen werden ausprobiert („funktionen.php“), welche mit Hilfe von GET- und POST-Methoden aufgerufen werden (in „tests.php“); das Menü wird um den Punkt „Tests“ („tests.php“) erweitert	gemeinsames Erarbeiten L+S	Beamer, PCs
15 Min.	*Vertiefungsphase*: eigene Funktionen erstellen (z.B. Flächen- und Umfangberechnungen, siehe „funktionen.php“)	Einzel- oder Partnerarbeit	Beamer, PC
20 Min.	*Festigungsphase*: Implementierung eines Logins („login.php“) mittels Session (um das Web Interface vor Fremdzugriff zu schützen)	gemeinsames Erarbeiten L+S	Beamer, PC
2-3 Min.	Aufgabe bis zur nächsten Einheit für Mitarbeitsplus: wer noch nicht fertig geworden ist → Übung zu Kontrollstrukturen und Funktionen fertig ausarbeiten; Ausblick auf kommende Einheit geben (Datenbanken); Verabschiedung	L-Gespräch	–

Tabelle 5 Planungsmatrix – Szenario 2

5.2.3 Ausarbeitung

Theorie

Die notwendige Theorie für dieses Unterrichtsszenario kann ebenfalls dem Kapitel „3.1 PHP“ entnommen werden – vor allem „3.1.4 Variablen“, „3.1.6 Funktionen“, „3.1.7 Kontrollstrukturen“, „3.1.8 Vordefinierte Informationen“ und „3.1.9 Einbinden externer Dateien“ sind relevant.

Beschreibung der Unterrichtsphasen

Hinführungsphase:

In dieser Phase wird den Lernenden von der Lehrperson die Theorie zu fortgeschrittenen Konzepten bezüglich PHP und dessen Einbindung in HTML nähergebracht. In diesem Zusammenhang sind besonders folgende Themen bedeutsam: Session, wichtige Variablen in Bezug auf Formularaufrufe (GET- und SET), Kontrollstrukturen und Funktionen. Gerade in Bezug auf Funktionen wird auch erläutert, was es heißt, „gut“ zu programmieren, d.h. hier geht es einerseits um eine besondere Übersicht, wozu Funktionen positiv beitragen können und andererseits um die Wiederverwendbarkeit von Code.

Erarbeitungsphase:

Es werden bestimmte Funktionen inklusive Kontrollstrukturen gemeinsam ausprobiert („funktionen.php“), welche mit Hilfe von GET- und POST-Methoden aufgerufen werden (in „tests.php“). Des Weiteren wird das Menü um den Punkt „Tests“ („tests.php“) erweitert.

Vertiefungsphase:

Als nächster Schritt – und zugleich als Vertiefung des soeben gelernten – sollen die Lernenden nun eigene Funktionen erstellen (in Einzel- oder Partnerarbeit), wie z.B. diverse Flächen- und Umfangberechnungen (siehe „funktionen.php“). Diese Übungen sollen dafür gut sein, das soeben erarbeitete Wissen bezüglich bestimmter Konzepte nicht nur zu reproduzieren, sondern auch erfolgreich anwenden zu können (Transferleistung). Auch der interdisziplinäre Aspekt kommt dabei nicht zu kurz, da es bei den Aufgaben auch darum geht, mathematische Konzepte (bspw. in Bezug auf Geometrie, Arithmetik, Logik) anzuwenden.

Während der Übungszeit steht die Lehrperson natürlich für auftretende Fragen oder Probleme zur Verfügung. Schüler und Schülerinnen, welche die Aufgaben bereits erfolgreich gelöst haben, sollen auch ihren Klassenkameradinnen und -kameraden helfen, wenn diese nicht mehr weiter wissen.

Das Fertigstellen dieser Übungen wird am Ende der Stunde als freiwillige Hausübung aufgegeben (für ein Mitarbeitsplus) und es wird den Lernenden gesagt, dass der Code zu

Beginn der nächsten Einheit präsentiert und besprochen werden wird (freiwillige Präsentationen für Mitarbeitsplus).

Festigungsphase:

In dieser Phase soll zur Festigung von bereits gehörten und ausprobierten Konzepten (Formularaufrufe und Kontrollstrukturen) gemeinsam mit den Schülerinnen und Schülern auf ihrem Web Interface ein Login mittels Session implementiert werden, um das Web Interface vor Fremdzugriff zu schützen.

5.2.4 Code-Beispiele

Folgende Dateien für Unterrichtsszenario 2 sind unverändert geblieben: „uebermich.php“, „index.php“ und „styles.css“ (trotzdem sind sie auf der CD-ROM im Ordner „Szenario 2“ enthalten). Weitere Dateien, in welchen Veränderungen vorgenommen wurden oder die benötigt werden und welche deshalb neu angelegt wurden, sind die folgenden:

Datei „menue.php“:

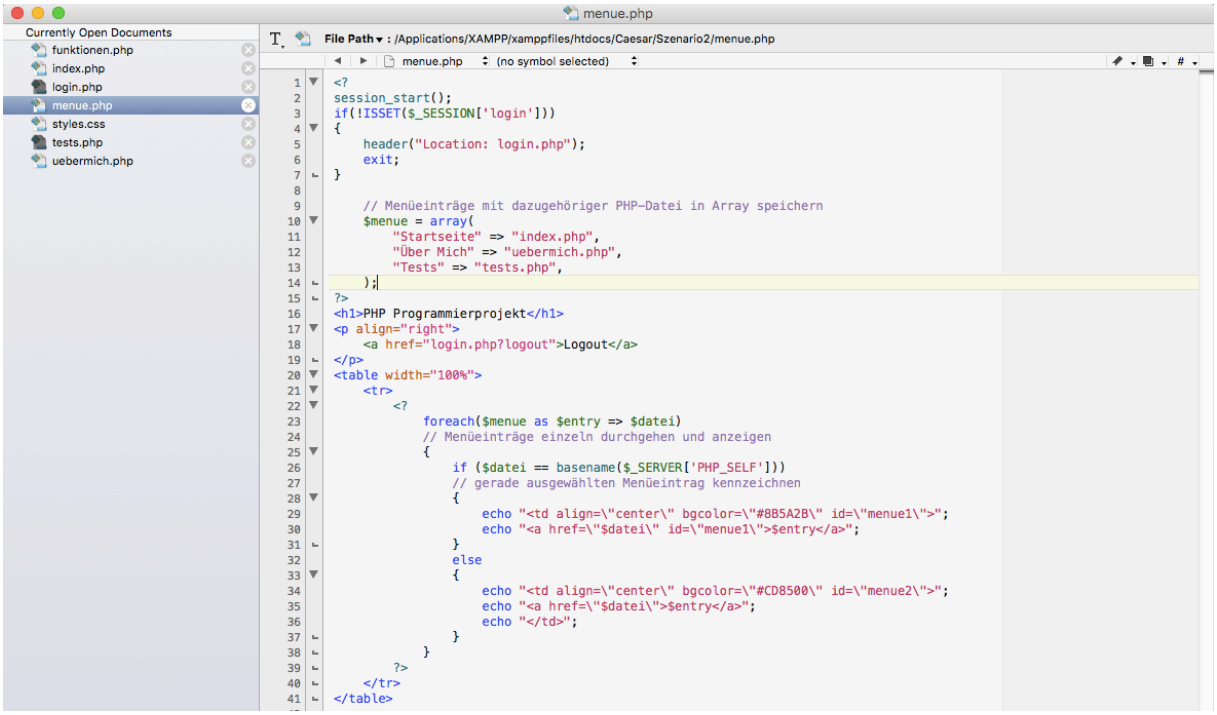


Abbildung 141 Szenario 2 – „menue.php“

Datei „login.php“:

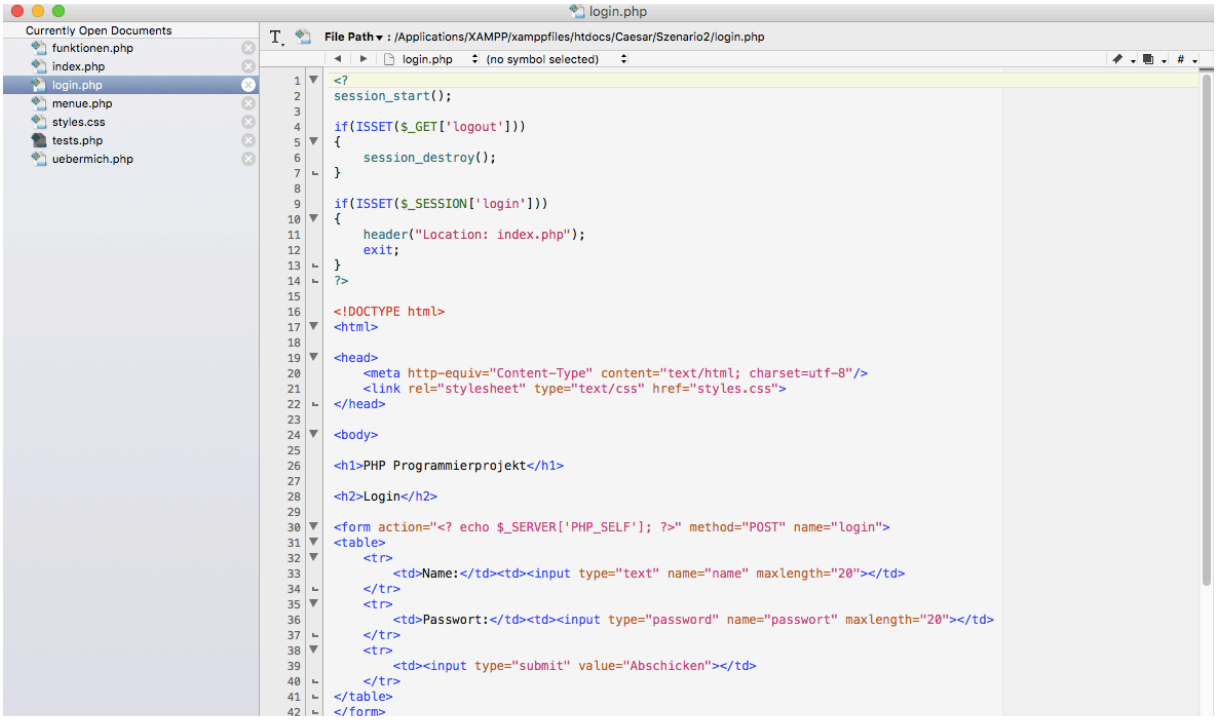


Abbildung 142 Szenario 2 – „login.php“ (Teil 1)

Abbildung 143 Szenario 2 – „login.php“ (Teil 2)

Datei „tests.php“:

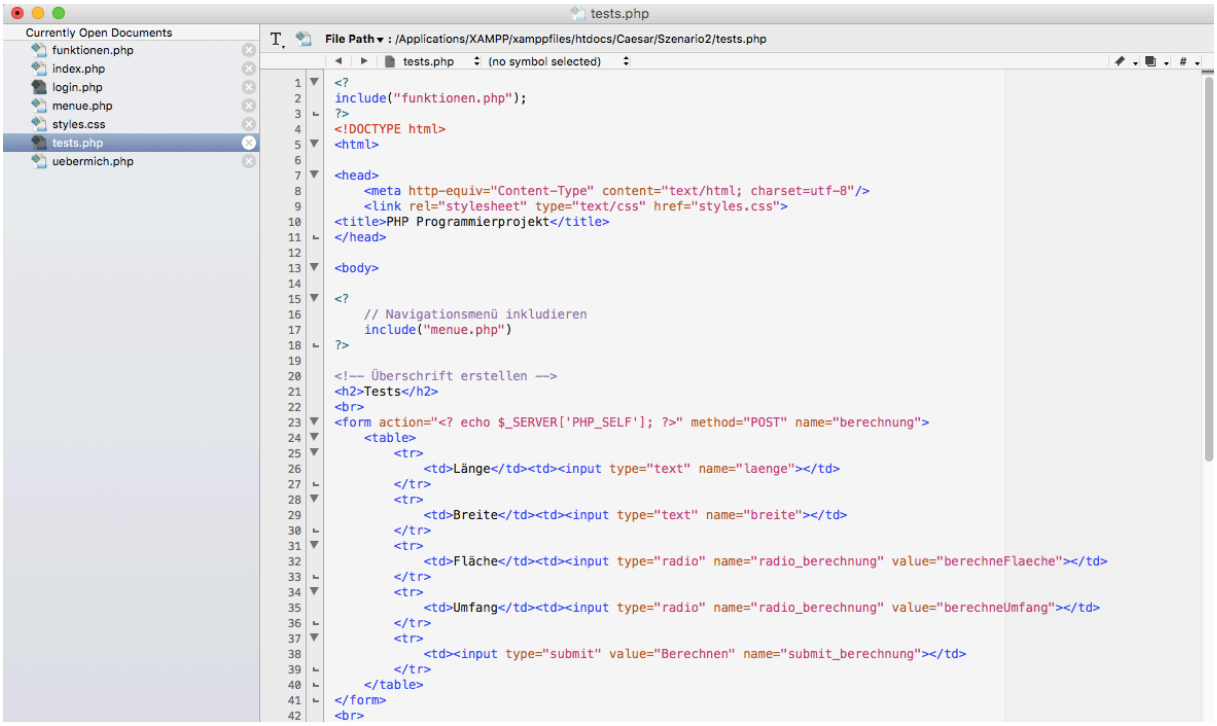


Abbildung 144 Szenario 2 – „tests.php“ (Teil 1)

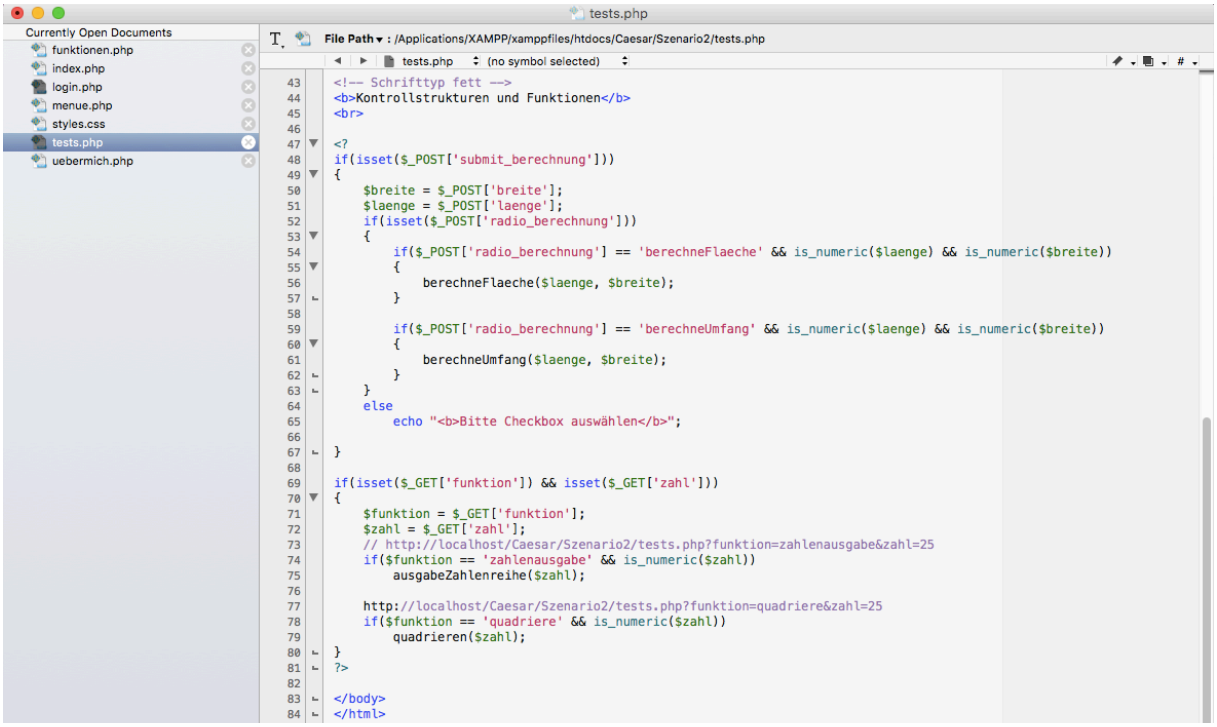


Abbildung 145 Szenario 2 – „tests.php“ (Teil 2)

Datei „funktionen.php“:

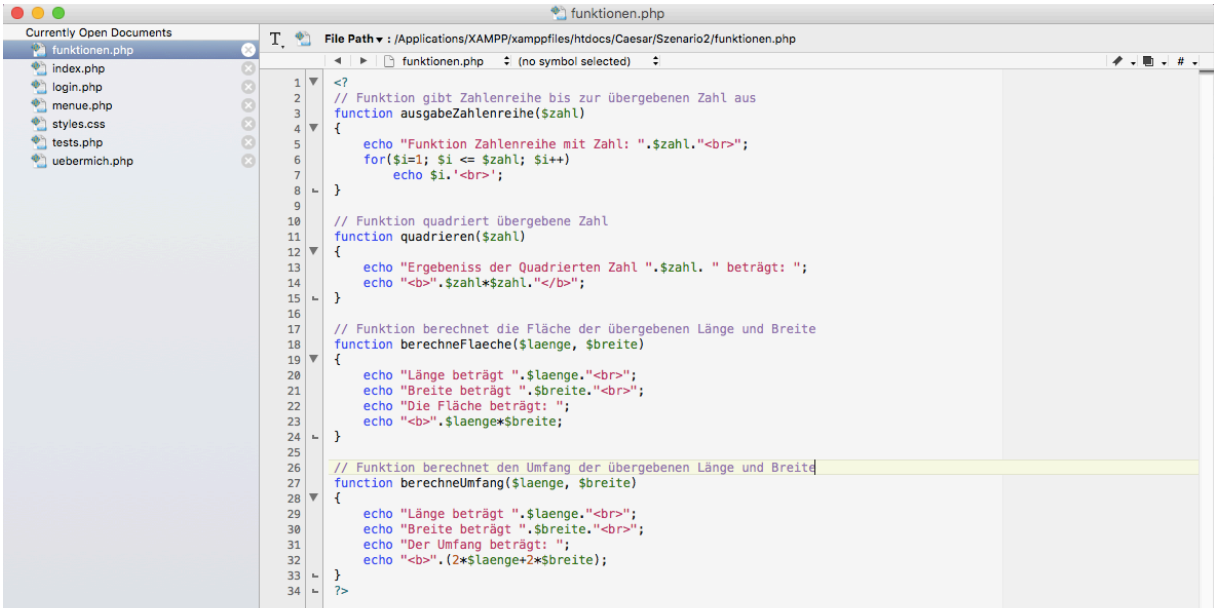


Abbildung 146 Szenario 2 – „funktionen.php“

5.3 Szenario 3: Datenbankgrundlagen und Einbindung in PHP-Code

5.3.1 Allgemeines

Thema: Datenbanken

Schulform: AHS Oberstufe

Lehrplanbezug:

Lehrstoff Wahlpflichtfach Informatik – 6. bis 8. Klasse[174]*:*

- Grundprinzipien der Informationsverarbeitung
- Konzepte von Betriebssystemen
- Aufbau und Funktionsweise von Netzwerken
- Datenbanken
- Lern- und Arbeitsorganisation
- Konzepte von Programmiersprachen
- künstliche Intelligenz
- Erweiterung der theoretischen und technischen Grundlagen der Informatik – grundlegende Algorithmen und Datenstrukturen
- Informatik, Gesellschaft und Arbeitswelt
- Rechtsfragen

[174] URL: https://www.bmbf.gv.at/schulen/unterricht/lp/lp_neu_ahs_21_11876.pdf?4dzgm2, aufgerufen am 2.12.2015.

Mit diesem Szenario wird offenkundig auf folgenden Aspekt des Lehrplans Bezug genommen: „Datenbanken“.

Angestrebte Stundenziele:

Wissenserweiterung und Kompetenzerwerb: Die Schülerinnen und Schüler sollen vielseitiges Wissen und Kompetenzen zum Thema Datenbanken erworben haben und dieses Wissen auch erfolgreich anwenden können. So sollen sie imstande sein, eine Datenbank mit Hilfe von phpMyAdmin anzulegen, darin Tabellen anlegen, diese mit Inhalten befüllen und Abfragen durchführen zu können. Des Weiteren sollen die Lernenden fähig sein, eine Datenbank-verbindung herzustellen und diverse SQL-Statements auszuführen (z.B. bestimmte Datensätze auswählen, Datensätze nach bestimmten Kriterien filtern, Datensätze einfügen und löschen).

Sozialverhalten: Die Schülerinnen und Schüler sollen lernen, Ergebnisse vor der Klasse zu präsentieren, weiters sollen sie aufmerksam zuhören können, wenn Klassenkolleginnen und -kollegen sprechen und auch konstruktives Feedback geben und annehmen können.

5.3.2 Planungsmatrix

Ablauf (Doppelstunde – 2 x 50 min):

Zeit	Inhalt und Phase	Sozialform	Material
2-3 Min.	warten, bis alle Schülerinnen und Schüler in der Klasse angekommen sind; Begrüßung	L-S-Gespräch(e)	–
10 Min.	freiwillige Präsentationen der Übungen der letzten Einheit, Code durchbesprechen, allfällige Fragen klären	L-S-Gespräch(e)	ev. Beamer, Lehrer-PC
15 Min.	*Hinführungsphase:* den Lernenden die Theorie zu Datenbanken näherbringen (wozu Datenbanken, Vorteile, SQL)	L-Vortrag	Beamer, Lehrer-PC, ev. Handouts
30 Min.	*Erarbeitungsphase*: gemeinsam mittels phpMyAdmin eine Datenbank anlegen, Tabellen erstellen, Testdaten eingeben; anschließend mittels PHP vom Web Interface aus eine Datenbankverbindung herstellen, SQL-Statements in Administrationsoberfläche der Datenbank (phpMyAdmin) erklären und testen	gemeinsames Erarbeiten L+S	Beamer, PCs
15 Min.	*Vertiefungsphase*: Übung: eigene Tabelle („tbl_book“) anlegen mit bestimmten Attributen („id“, „title“, „author“, „genre“, „year“, „lent“ als Boolean), mit Datensätzen befüllen, bestimmte SQL-Statements darauf anwenden	Einzelarbeit	PCs

25 Min.	*Festigungsphase:* Abfragen mittels SQL-Statements – Implementierung in PHP, einbinden von „funktionen.php“ ins Menü Übung: SQL-Statements in PHP auf eigene Tabellen anwenden	gemeinsames Erarbeiten L+S, Einzelarbeit	Beamer, PCs
2-3 Min.	freiwillige HÜ (Übung fertig ausarbeiten); Ausblick auf kommende Einheit geben (Kryptographie); Verabschiedung	L-Gespräch	–

Tabelle 6 Planungsmatrix – Szenario 3

5.3.3 Ausarbeitung

Theorie

Die notwendige Theorie für dieses Unterrichtsszenario kann dem Kapitel „3.2 Datenbank“ entnommen werden. Weiters ist das Kapitel „3.1 PHP“ relevant in Bezug auf die Programmierung – hier sind vor allem die Unterkapitel „3.1.7 Kontrollstrukturen“, „3.1.8 Vordefinierte Informationen“ und „3.1.9 Einbinden externer Dateien“ wichtig.

Beschreibung der Unterrichtsphasen

Hinführungsphase:

Den Lernenden wird von der Lehrperson die Theorie zu Datenbanken nähergebracht, insbesondere wozu Datenbanken benötigt werden und welche Vorteile sie haben. Als Einstieg kann mittels Brainstorming zu Datenbanken an das Vorwissen der Schülerinnen und Schüler zur Sprache gebracht und daran angeknüpft werden.

Erarbeitungsphase:

In dieser Phase wird gemeinsam mit den Schülerinnen und Schülern eine Datenbank mit Hilfe von phpMyAdmin angelegt. Des Weiteren wird eine Tabelle erstellt und mit Datensätzen befüllt (siehe Abb. 147). Auch gewisse SQL-Statements sollen gemeinsam ausprobiert werden – in einem ersten Schritt in phpMyAdmin, später (siehe Festigungsphase) in PHP-Code. In einem nächsten Schritt soll dann eine Datenbankverbindung vom Web Interface zur Datenbank hergestellt werden (siehe „db.php“).

Vertiefungsphase:

Als Nächstes folgt – als Vertiefung des soeben gelernten – eine Übung, die von den Lernenden in Einzelarbeit durchgeführt werden soll. So soll nämlich eine weitere Tabelle, nämlich „tbl_book“ angelegt werden mit bestimmten Attributen („id“, „title“, „author“, „genre“, „year“, „lent“ als Boolean), diese mit individuellen Datensätzen befüllt werden und

anschließend bestimmte SQL-Statements darauf angewendet werden. Diese Aufgaben dienen zur Reproduktion und Anwendung von soeben erworbenem Wissen und erlernten Fertigkeiten (Transferleistung).

Festigungsphase:

In dieser Phase wird den Lernenden von der Lehrperson gezeigt, wie bestimmte Abfragen (SQL-Statements) in PHP implementiert werden können. Dann werden diverse Abfragen (siehe „database.php“) durchgeführt und deren Ergebnisse ausgegeben. Diese neue Datei soll auch in das Menü eingebunden werden. Als kleine abschließende Übung sollen diverse SQL-Statements nun in PHP auf die eigens erstellte Tabelle „tbl_book“ angewendet werden.

5.3.4 Code-Beispiele

Folgende Datenbanktabelle „tbl_person“ wurde für Szenario 3 eigens angelegt und wurde in nachfolgendem Code verwendet:

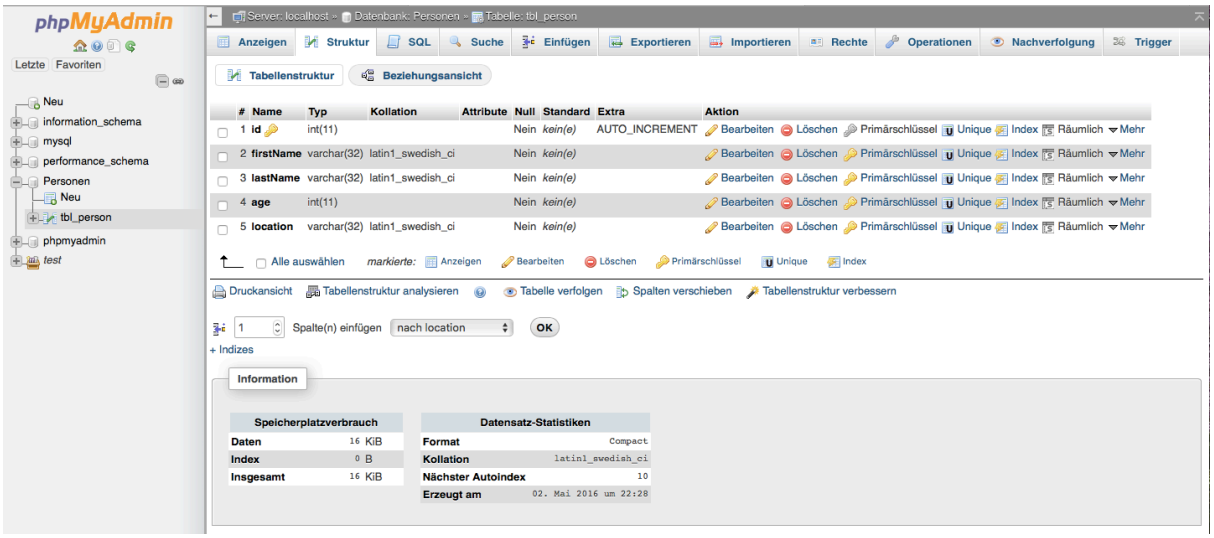


Abbildung 147 Szenario 3 – Datenbanktabelle

Die folgenden Dateien wurden neu angelegt bzw. inhaltlich ergänzt: „db.php“, „database.php“ und „menue.php“.

Datei „db.php“:

Abbildung 148 Szenario 3 – „db.php“

Datei „database.php“:

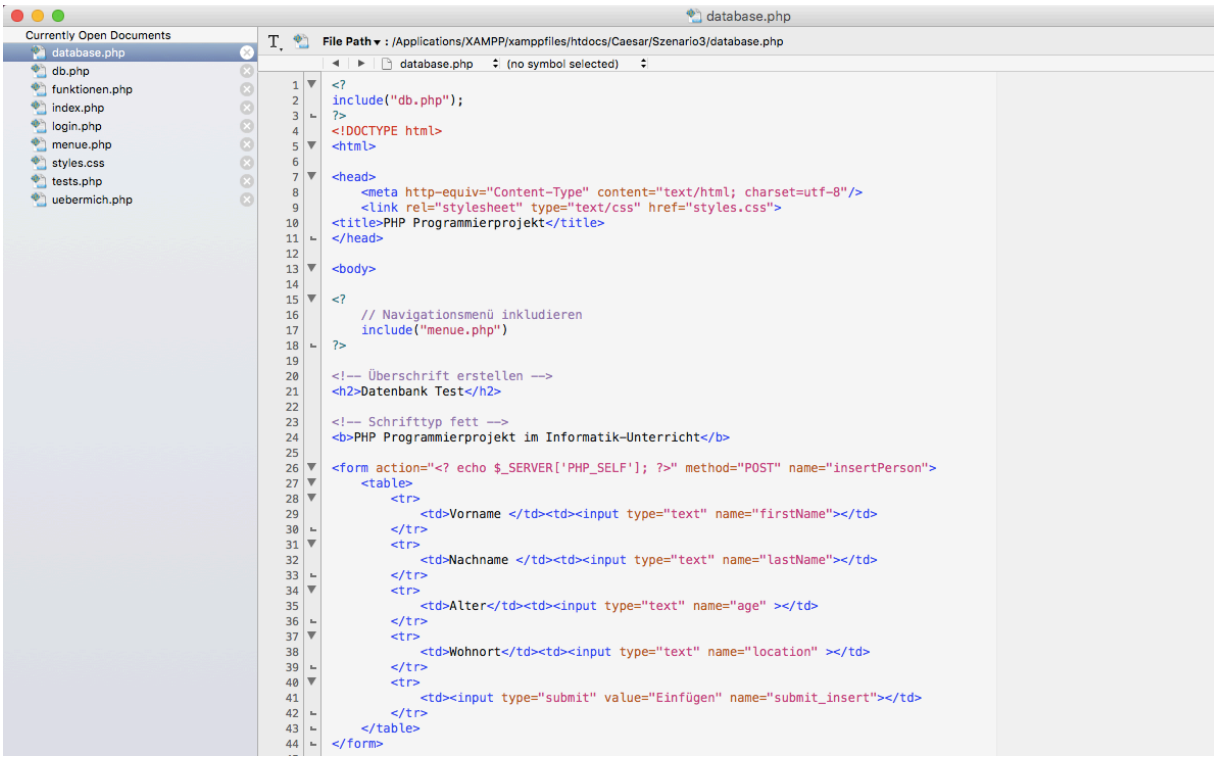


Abbildung 149 Szenario 3 – „database.php“ (Teil 1)

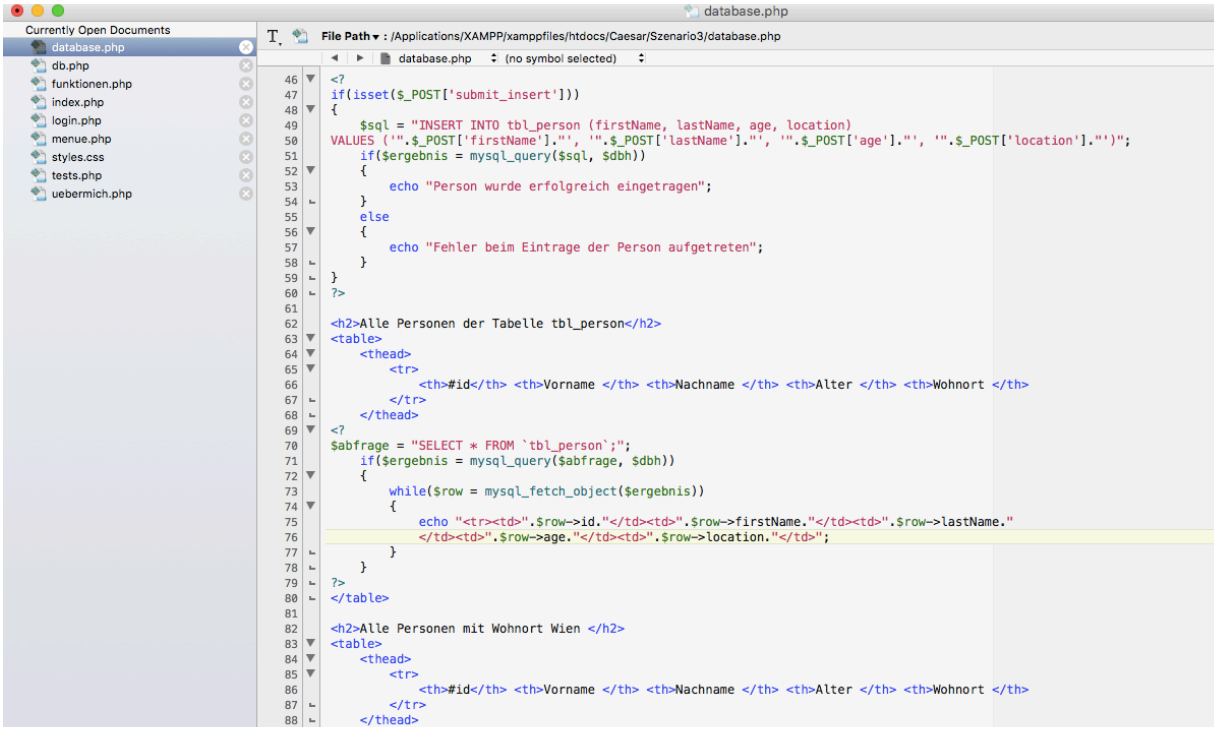


Abbildung 150 Szenario 3 – „database.php“ (Teil 2)

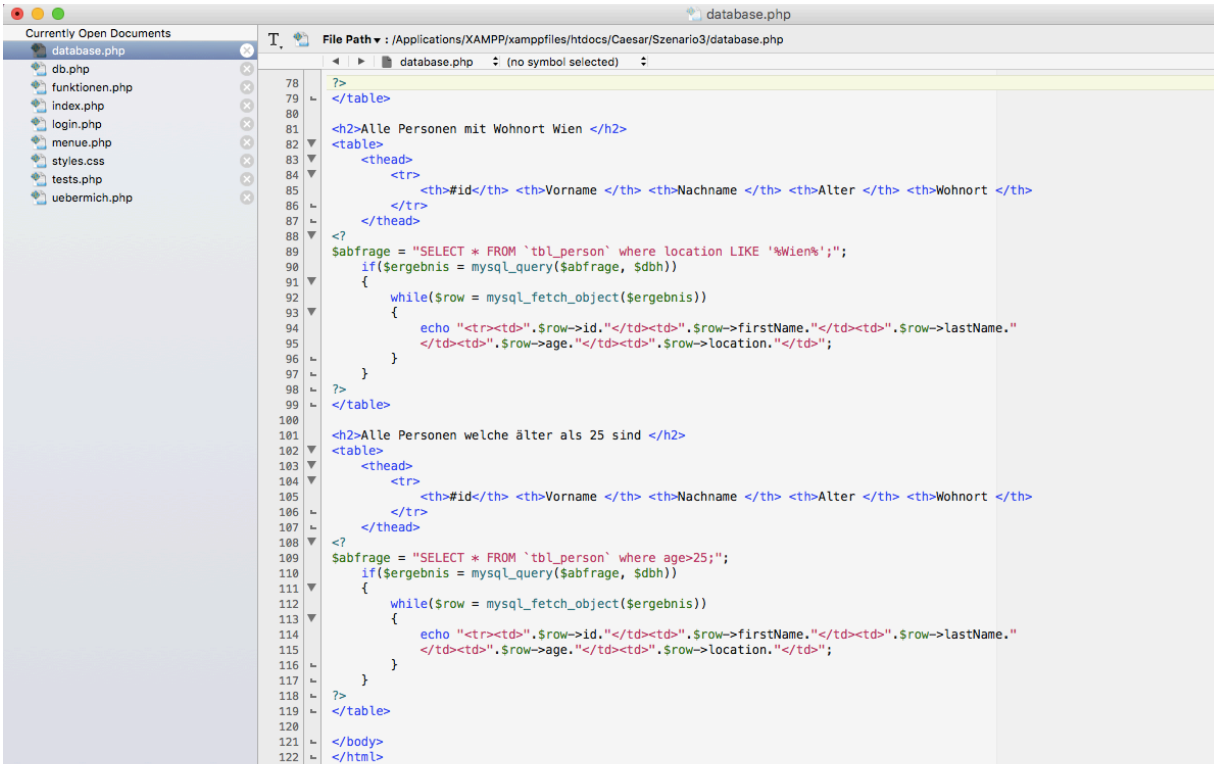


Abbildung 151 Szenario 3 – „database.php“ (Teil 3)

Datei „menue.php“:

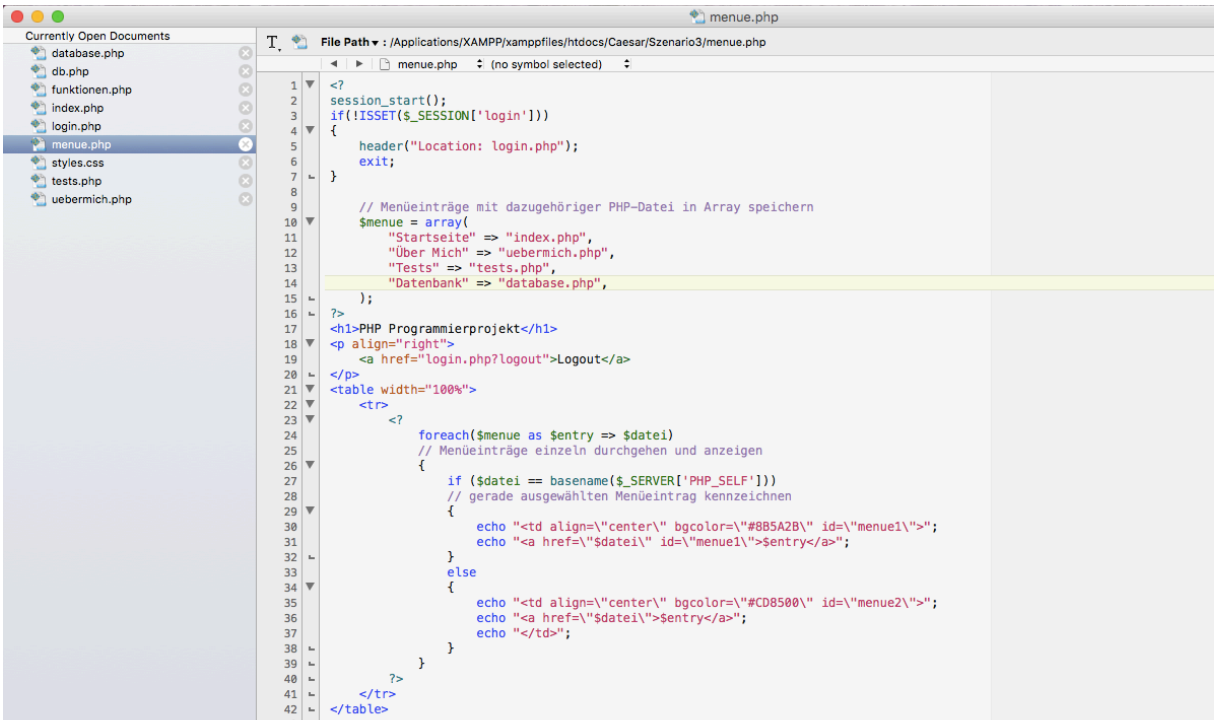


Abbildung 152 Szenario 3 – „menue.php“

5.4 Szenario 4: Kryptographie: Theorie und Praxis

5.4.1 Allgemeines

Thema: Kryptographie, PHP

Schulform: AHS Oberstufe

Lehrplanbezug:

Lehrstoff Wahlpflichtfach Informatik – 6. bis 8. Klasse[175]*:*

- Grundprinzipien der Informationsverarbeitung
- Konzepte von Betriebssystemen
- Aufbau und Funktionsweise von Netzwerken
- Datenbanken
- Lern- und Arbeitsorganisation
- Konzepte von Programmiersprachen
- künstliche Intelligenz
- Erweiterung der theoretischen und technischen Grundlagen der Informatik – grundlegende Algorithmen und Datenstrukturen
- Informatik, Gesellschaft und Arbeitswelt
- Rechtsfragen

Mit diesem Szenario wird auf die folgenden zwei Aspekte des Lehrplans Bezug genommen: „Erweiterung der theoretischen und technischen Grundlagen der Informatik – grundlegende Algorithmen und Datenstrukturen“ und „Konzepte von Programmiersprachen“.

Angestrebte Stundenziele:

Wissenserweiterung und Kompetenzerwerb: Die Schülerinnen und Schüler sollen vielseitiges Wissen und Kompetenzen zum Thema Kryptographie erworben haben – sowohl in der Theorie als auch in der Praxis. So sollen die Lernenden beispielsweise wissen, was man unter dem Begriff Kryptographie versteht, wozu Verschlüsselung gut ist. Des Weiteren sollen sie imstande sein, die Funktionsweise der Cäsar-Verschlüsselung beschreiben, sie in PHP implementieren und auch Veränderungen am Code durchführen zu können (Transferleistung).

Sozialverhalten: Die Schülerinnen und Schüler sollen lernen, Ergebnisse vor der Klasse zu präsentieren, weiters sollen sie aufmerksam zuhören können, wenn Klassenkolleginnen und -kollegen sprechen und auch konstruktives Feedback geben und annehmen können.

[175] URL: https://www.bmbf.gv.at/schulen/unterricht/lp/lp_neu_ahs_21_11876.pdf?4dzgm2, aufgerufen am 2.12.2015.

5.4.2 Planungsmatrix

Ablauf (Doppelstunde – 2 x 50 min):

Zeit	Inhalt und Phase	Sozialform	Material
2-3 Min.	warten, bis alle Schülerinnen und Schüler in der Klasse angekommen sind; Begrüßung	L-S-Gespräch(e)	–
10 Min.	freiwillige Präsentationen der HÜ bzw. Musterlösung besprechen	Präsentation	Beamer, Lehrer-PC
20 Min.	*Hinführungsphase:* Theorie zu Kryptographie, u.a. Caesar-Verschlüsselung, Verschlüsselung bei den Zahlenrätseln von Ephesos	L-Vortrag	Beamer, Lehrer-PCs, ev. Handouts
20 Min.	*Erarbeitungsphase*: Code zum Verschlüsseln eines beliebigen, eingegebenen Textes gemeinsam erarbeiten (siehe Code-Beispiele) und testen; gemeinsame Fehlersuche; einbinden von „krypto.php“ ins Menü	gemeinsames Erarbeiten L+S	Beamer, PCs
20 Min.	*Vertiefungsphase*: Code zum Entschlüsseln eines beliebigen, eingegebenen Textes selbstständig oder in Partnerarbeit erarbeiten lassen	Einzel- oder Partnerarbeit	PCs
5 Min.	*Festigungsphase:* den erarbeiteten, funktionierenden Code (siehe Code-Beispiele) per Beamer herzeigen und durchbesprechen; falls das Programm bei manchen S noch nicht funktioniert → bei Fehlersuche helfen	L-S-Gespräch	Beamer, PCs
20 Min.	S sollen sich eine geheime Botschaft/Rätsel überlegen, dann verschlüsseln (mit geheimem Schlüssel), ausdrucken und L abgeben → L mischt alle Zettel und teilt jeweils einen Zettel einem je Lernenden aus; Aufgabe: Botschaft entschlüsseln → z.B. zuerst händisch, danach digital mittels selbst programmiertem Entschlüsselungsprogramm → Schlüssel muss dafür natürlich herausgefunden werden, da dieser nicht bekannt ist (trial-and-error-Prinzip)	Einzelarbeit	geheime Botschaften/ Rätsel ausgedruckt auf Zettel
2-3 Min.	freiwillige Aufgabe bis zur nächsten Einheit: Rätsel entschlüsseln (falls noch nicht in Stunde geschafft); Ausblick auf kommende Einheit geben (möglicherweise Datenschutz und Datensicherheit); Verabschiedung	L-Gespräch	–

Tabelle 7 Planungsmatrix – Szenario 4

5.4.3 Ausarbeitung

Theorie

Die notwendige Theorie für dieses Unterrichtsszenario kann dem Kapitel „3.4 Kryptographie“ entnommen werden – vor allem das Unterkapitel „3.4.2 Einfache Verschlüsselungsmethoden“ ist relevant für die Programmierung. Des Weiteren sind folgende Unterkapitel des „3.1 PHP“-Kapitels wichtig: „3.1.4 Variablen“, „3.1.7 Kontrollstrukturen“ und „3.1.8 Vordefinierte Informationen“.

Beschreibung der Unterrichtsphasen

Hinführungsphase:

Diese Phase ist dazu da, um den Schülerinnen und Schülern einen gewissen Grundstock an Theoriewissen zu den Themen der heutigen Einheit (Kryptographie allgemein, Caesar-Verschlüsselung, Verschlüsselung bei den Zahlenrätseln von Ephesos) zu vermitteln. Wie genau diese Theorievermittlung umgesetzt wird – beispielweise mittels eines Lehrpersonvortrages oder interaktiv mittels Internetrecherche – möchte ich hiermit nicht eindeutig festlegen, sondern offenlassen für eigene Ideen. Als Einstieg kann mittels eines Brainstormings zu Verschlüsselung das Vorwissen der Schülerinnen und Schüler zur Sprache gebracht und daran angeknüpft werden. Wichtig ist es auch, Sicherheits- und Datenschutzaspekte anzusprechen (so gibt es bspw. bereits seit kurzem auch schon Ende-zu-Ende-Verschlüsselung bei WhatsApp → Bezug zur Lebenswelt der Lernenden herstellen; Thema „Datenschutz und Datensicherheit“ als mögliche Folgeeinheit).

Erarbeitungsphase:

Der Schwerpunkt dieser Phase liegt auf der Implementierung der Cäsar-Verschlüsselung in PHP (siehe „krypto.php“). Insofern wird der PHP-Code zum Verschlüsseln eines beliebigen, in ein Formularfeld eingegebenen Textes gemeinsam mit den Schülerinnen und Schülern erarbeitet (zusätzliches Formularfeld „Schlüssel“, um welches der Text verschoben wird, wie dies bei der Cäsar-Verschlüsselung der Fall ist). Hierbei spielt natürlich auch immer das Testen eine große Rolle und die gemeinsame Fehlersuche, wenn etwas nicht (so) funktioniert, wie es soll. Die neue Datei „krypto.php“ wird anschließend ins Menü eingebunden.

Vertiefungsphase:

Diese Phase ist dazu da, den Code zum Entschlüsseln eines beliebigen, in ein Formularfeld eingegebenen Textes selbstständig oder in Partnerarbeit erarbeiten zu lassen. Diese Phase baut auf die vorherige insofern auf, als dass der erstellte Code von den Lernenden verstanden

werden muss, um das erzeugte Programm derartig abändern zu können, dass es in die „umgekehrte“ Richtung funktioniert (Transfer- und Problemlösungskompetenz neben Reproduktion). Wissen die Schülerinnen und Schüler nicht weiter bzw. treten Fehler auf, welche von den Lernenden nicht von selbst gelöst werden können, so werden diese von der Lehrperson oder Klassenkameradinnen und -kameraden mit Tipps versorgt.

Festigungsphase:

In dieser Phase wird der soeben erarbeitete Code gemeinsam durchbesprochen – entweder in einem Lehrperson-Schülerinnen/Schüler-Gespräch oder als Erklärung von einzelnen ausgewählten Lernenden. Dabei wird der funktionierende Code per Beamer hergezeigt. Falls das Programm bei manchen Schülerinnen und Schülern immer noch nicht funktioniert, wird bei der Fehlersuche geholfen.

Anschließend startet der kreative Teil dieser Unterrichtseinheit: So sollen die Lernenden sich nämlich eine geheime Botschaft oder ein Rätsel überlegen (oder eines im Internet suchen), dieses dann mit Hilfe des Verschlüsselungsprogramms verschlüsseln (mit einem geheimen Schlüssel), ausdrucken und der Lehrperson abgeben. Die Lehrerin bzw. der Lehrer mischt dann alle Zettel durch und teilt jeweils einen Zettel je einer bzw. einem Lernenden aus. Die Aufgabe besteht dann darin, die Botschaft zu entschlüsseln, z.B. zuerst händisch, dann mittels selbst programmiertem Entschlüsselungsprogramm – der Schlüssel muss dafür natürlich herausgefunden werden, da dieser nicht bekannt ist (trial-and-error-Prinzip).

5.4.4 Code-Beispiele

Für Unterrichtsszenario 4 wird zusätzlich eine neue Datei benötigt („krypto.php“) und eine weitere Datei wird ergänzt („menue.php“).

Datei „krypto.php“:

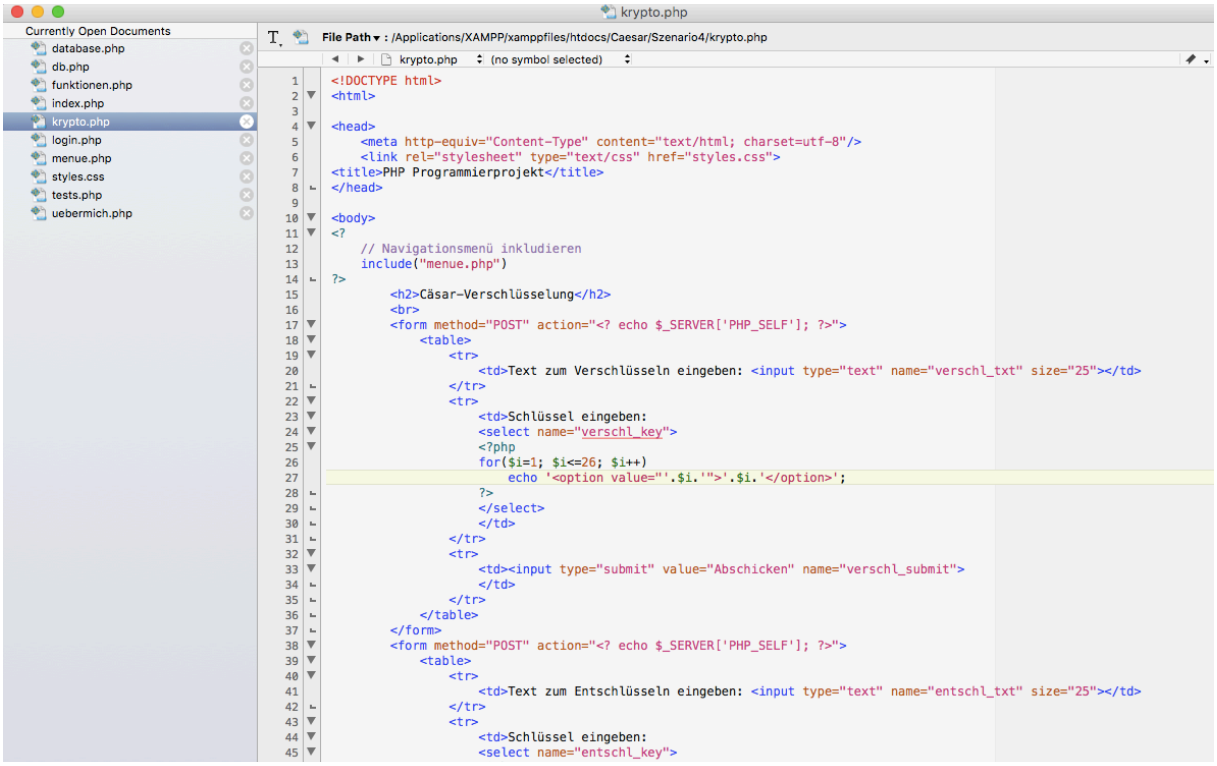


Abbildung 153 Szenario 4 – „krypto.php“ (Teil 1)

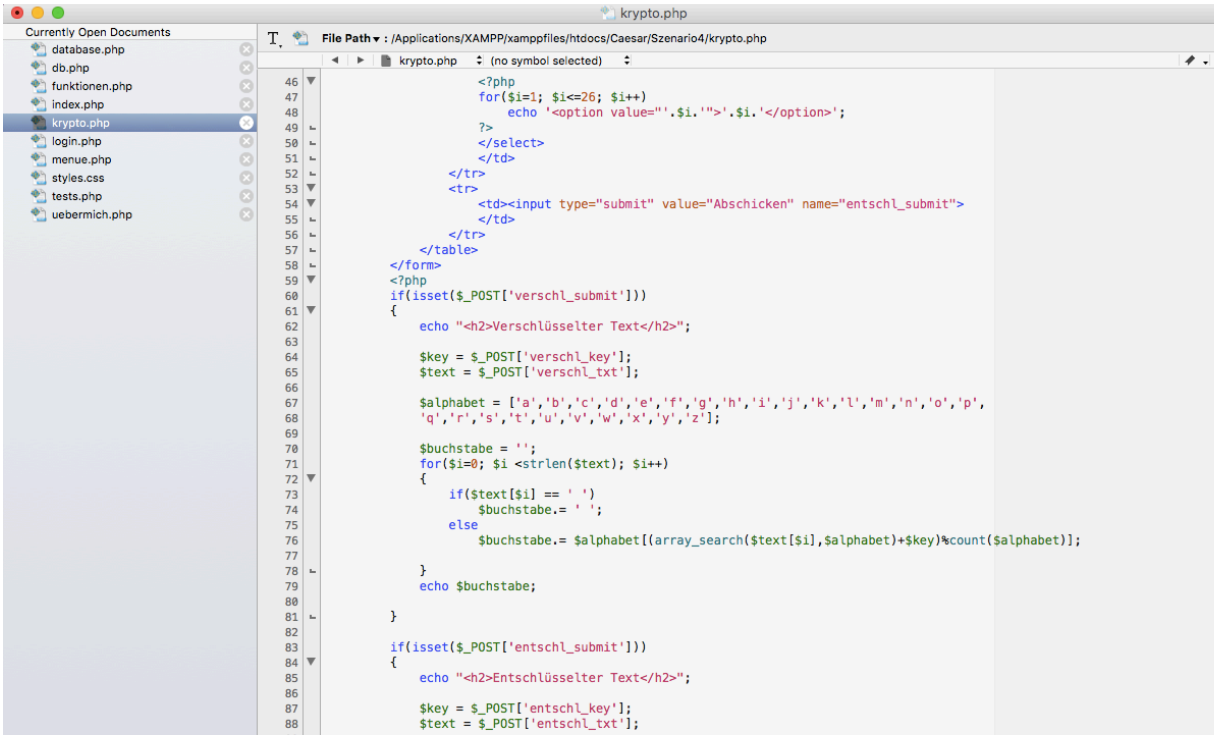


Abbildung 154 Szenario 4 – „krypto.php“ (Teil 2)

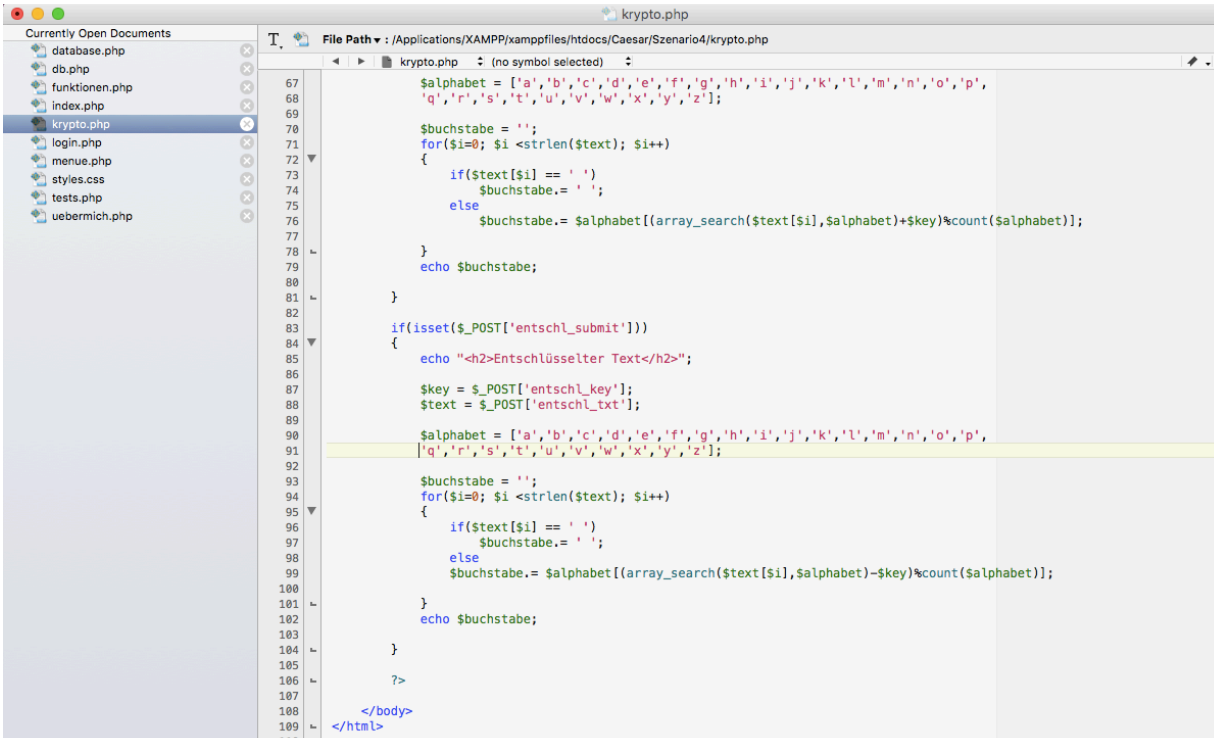


Abbildung 155 Szenario 4 – „krypto.php“ (Teil 3)

Datei „menue.php“:

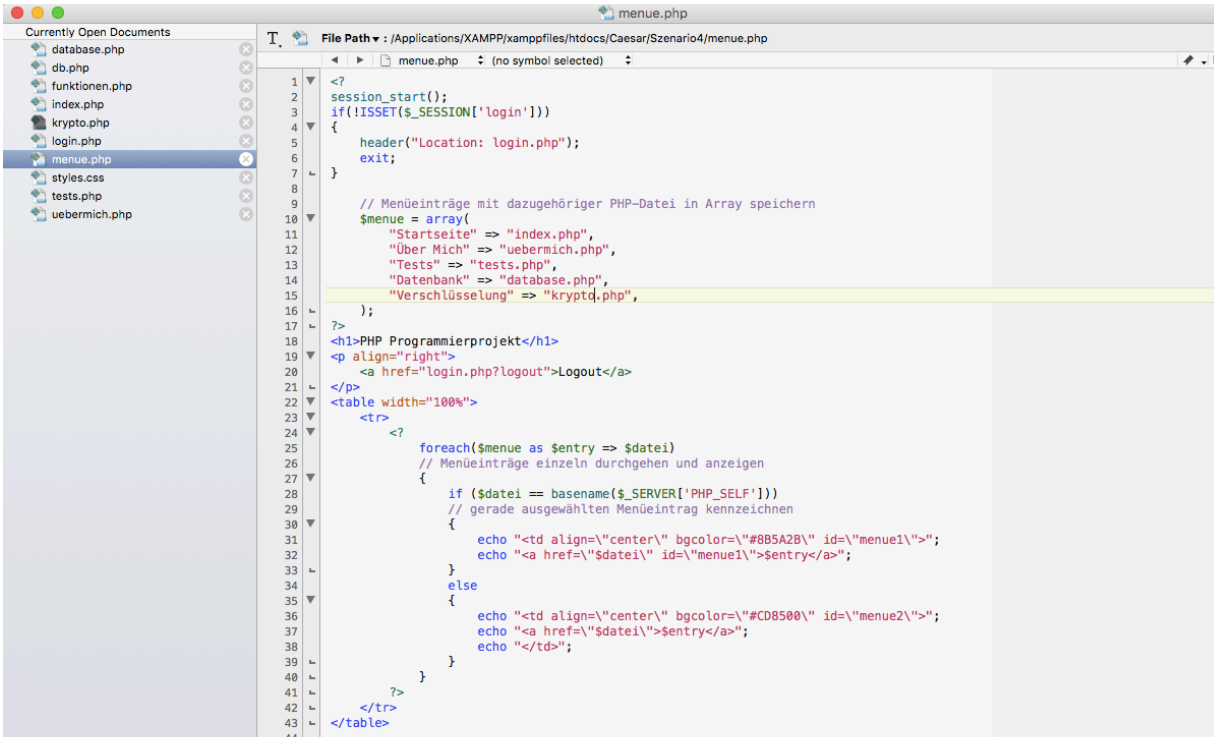


Abbildung 156 Szenario 4 – „menue.php“

6. Geschichtlicher Hintergrund

Dieses Kapitel befasst sich mit dem geschichtlichen Hintergrund dieser Arbeit – also insofern mit der Stadt Ephesos, welche maßgeblich für diese Diplomarbeit ist, da es ja beim Programmierprojekt um die Entschlüsselung der Zahlenrätsel aus Ephesos geht. Ohne diese hätte sich dieser Teil der Diplomarbeit gar nicht erst entwickeln können. Dementsprechend ist dieser antiken Metropole ein eigenes Unterkapitel gewidmet. Darin wird zuerst allgemein über Ephesos berichtet und über die geographische Lage dieser Stadt. Dann wird ein geschichtlicher Überblick gegeben und schließlich auf besondere berühmte Baudenkmäler eingegangen. Das darauffolgende Unterkapitel nimmt die Hanghäuser von Ephesos genauer unter die Lupe, in welchen sich besagte Zahlenrätsel und natürlich auch weitere Inschriften und archäologische Fundstücke finden haben lassen. Im letzten Unterkapitel geht es schließlich um Inschriften und Epigraphik allgemein (konkret um die Zahlenrätsel von Ephesos geht es hingegen im Kapitel „4.1 Was sind die Zahlenrätsel von Ephesos?“).

6.1 Ephesos

6.1.1 Allgemeines

Abbildung 157 Die Überreste des einstigen Weltwunders

Mit dem Stadtnamen „Ephesos“ assoziiert man möglicherweise sofort den Tempel der Artemis, welcher eines der Sieben Weltwunder der Antike war und mit einer großen Faszination verbunden ist. Doch diese Stadt steht nicht nur für den Tempel, sondern war vielmehr eine der großen Metropolen der alten Welt, die es schaffte, über Jahrhunderte hinweg ihre Bedeutung zu wahren. Dies kann man auch heutzutage noch an dem großen Ausgrabungsgelände erkennen, welches seit mehr als 100 Jahren durch das Österreichische Archäologische Institut erforscht wird und wo es auch noch in Zukunft viel zu erforschen geben wird.[176]

Die einstige Größe der Stadt belegen nicht nur die zahlreichen Bauten, sondern auch die Menschen, die das städtische Klima hervorgebracht hat. Berühmte Persönlichkeiten, die in diesem Zusammenhang zu nennen sind, wären z.B. der Philosoph Heraklit (spätes 6./frühes 5. Jh. v. Chr., bekannt für seinen Spruch „alles fließt“) und der Geograph Artemidoros (1. Jh. v. Chr.).[177]

Zu Besuch in Ephesos waren auch mehrere bedeutende Persönlichkeiten, wie beispielsweise G. Iulius Caesar im Jahr 48 v. Chr. und die Caesarmörder M. Iunius Brutus und C. Cassius Longinus. Weiters hielten sich im Jahr 41 v. Chr. Marcus Antonius (82-30 v. Chr.) und Kleopatra VII. (Königin 52-30 v. Chr.) in der Stadt auf.[178] Der Apostel Paulus lebte im Jahr 52 während seiner zweiten Missionsreise in Ephesos und nach christlicher Tradition fand Maria hier ihren letzten Wohnsitz.[179]

Wichtig ist noch anzumerken, dass Ephesos in erster Linie eine Handels- und Verwaltungsstadt war, deren bedeutendster Erwerbszweig das Artemision war (siehe „6.1.4 Baudenkmäler und Funde“). Und auch die Tatsache, dass Ephesos immer wieder im Brennpunkt der Geschichte stand, wie z.B. in Bezug auf kriegerische Kontexte oder als Zentrum von nachhaltigen Geistesströmungen, wie beispielsweise dem Christentum, ist von großer Bedeutung und wird dementsprechend in einem späteren Kapitel („6.1.3 Geschichtlicher Überblick“) noch ausführlicher erörtert. [180]

[176] vgl. Letzner 2010, S. 7.
[177] vgl. Letzner 2010, S. 7.
[178] vgl. Letzner 2010, S. 13.
[179] vgl. Letzner 2010, S. 14.
[180] vgl. Letzner 2010, S. 7.

6.1.2 Geographische Lage

Abbildung 158 Geographische Lage von Ephesos

Ephesos liegt ca. 75 km südlich der heutigen türkischen Großstadt İzmir (siehe Abb. 158). Ursprünglich besaß die Stadt einen unmittelbaren Zugang zum Meer durch einen Hafen an der Mündung des Kaystros, der heute Küçük Menderes (Kleiner Mäander) genannt wird. Dabei handelte es sich um eine tief einschneidende Meeresbucht, die für antike Seefahrer gut geschützte Ankermöglichkeiten bot. Durch die Ablagerungen des Flusses ist der Hafen aber heute verlandet. Auch die von Süden kommende Flüsse namens Marnas und Selenos trugen

mit gewaltigen Schlammmassen zur Veränderung der Landschaft und zu diesem Prozess bei. Die Verlandung hat bereits in der Antike massive Probleme bereitet. Die Verlandung des Hafens und die Verschiebung der Küstenlinie wurde durch Meeresströmungen unterstützt, welche zur Bildung einer Nehrung (einer Landzunge vor der Küste) beitrugen. Insgesamt verschob sich die Küstenlinie um ca. 9 km nach Westen.[181]

Drei Hügel prägen in Ephesos das Stadtgebiet: der sogenannte Ayasoluk, welcher ursprünglich unmittelbar an der Meeresbucht lag und den in die Bronzezeit zurückreichenden Siedlungskern von Ephesos bildete; weiters der sogenannte Panayır Dağ, welcher ursprünglich die Küste bildete, und der durch einen Taleinschnitt abgetrennte Bülbül Dağ.[182] Alle drei sind auf der folgenden Abbildung ersichtlich:

Abbildung 159 Plan von Ephesos mit den drei Hügeln (v.l.n.r. Bülbül Dağ, Panayır Dağ und Ayasoluk

Die hellenistische Stadt füllte vermutlich nur die hafennahen Teile dieses riesigen Areals aus. Denn erst unter Kaiser Augustus, der Ephesos zur Hauptstadt der Provinz Asia erhob, oder den Jahrzehnten unmittelbar davor, erweiterte sich das städtisch verbaute Areal. Dabei wurde der Sattel zwischen den beiden Stadtbergen mit dem römerzeitlichen Regierungsviertel verbaut – die nächsten Generationen erlebten die bauliche Ausgestaltung der Talfurche zur Hafenebene hinab.[183]

Diese soeben angesprochenen topographischen Bedingungen waren u.a. dafür verantwortlich, dass Ephesos entstehen und gedeihen konnte und schließlich untergehen sollte, wie dies noch genauer im folgenden Unterkapitel („6.1.3 Geschichtlicher Überblick“) gezeigt werden wird.

[181] vgl. Letzner 2010, S. 9.
[182] vgl. Letzner 2010, S. 9.
[183] vgl. Scherrer 1995, S. 10.

6.1.3 Geschichtlicher Überblick

Prähistorie

Die ältesten Siedlungsspuren im Raum Ephesos reichen bis in das 5. Jahrtausend v. Chr. zurück. Aus der ab etwa 3600 v. Chr. beginnenden älteren Bronzezeit sind vor einigen Jahren auf dem Johanneskirchenberg nördlich unterhalb der byzantinischen Festung Keramikfunde aufgetaucht, welche eine Besiedelung in dieser Epoche beweisen. Der Berg war damals eine steil aufragende Insel in einer weit nach Osten reichenden, seichten Meeresbucht – somit ein idealer, gut zu verteidigender Platz, der rundum Sicherheit bot. Deutlicher wird diese Geschichte in der späteren Bronzezeit (insbesondere ab der 2. Hälfte des 2. Jahrtausends v. Chr.): Da entstanden nämlich an der Künste West- und Südwestkleinasiens minoische Stützpunkte, welche nach der Übernahme der altkretischen Macht und Kultur durch die mykenischen Griechen zu mykenischen Plätzen wurden (etwa nach 1450 v. Chr.). [184]

In der 2. Hälfte des 14. Jhs. v. Chr. findet man Anzeichen der soeben angesprochenen Griechen etwa mit einem mykenischen Grab auf dem Ayasoluk-Hügel. Erst während der sogenannten Ionischen Wanderung, die in Griechenland durch das Eindringen der Dorer ausgelöst wurde, kam es schließlich im 11. Jh. v. Chr. zu der Anlage einer dauerhaften Siedlung.[185]

Artemis Tempel und Heiligtum

Ein weiteres wichtiges Datum für die Entwicklung der Stadt lässt sich im 9. Jh. v. Chr. finden, in dessen Verlauf der Bezirk der Artemis entstanden ist. Dem Heiligtum kam auch eine politische Bedeutung zu, da der ephesische Tyrann Pythagoras der Göttin in der 2. Hälfte des 7. Jhs. v. Chr. einen Tempel stiftete. Das Artemisheiligtum sollte mit seiner Asylie (ein Privileg, das Schutz vor gewaltsamen Überfällen garantiert) für Ephesos von zentraler wirtschaftlicher Bedeutung sein. So konnten nämlich erhebliche Gebühren eingenommen werden für Deponierung von Gütern, welche als Weihung deklariert wurden. Weiters entstand auch eine beachtliche Souvenirindustrie, die der Stadt einen erheblichen Wohlstand brachte.[186]

Herrschaftsformen und weiterer Verlauf

Ephesos hat – wie fast alle griechischen Stadtstaaten – mehrere, für die griechische Geschichte typische Herrschaftsformen erlebt. Zuerst wurde Ephesos von den Nachkommen des Androklos, den Basiliden, regiert und in weiterer Folge gab es Oligarchien und

[184] vgl. Knibbe 1998, S. 59.
[185] vgl. Letzner 2010, S. 10.
[186] vgl. Letzner 2010, S. 10.

Tyrannenherrschaften (wie z.B. jene von Pythagoras, wie soeben angesprochen). Um 700 v. Chr. hat sich Ephesos als Mitglied des Ionischen Bundes mit dem Ziel der Ausdehnung seines Einflusses nach Süden an der Zerstörung der am Nordfuß des Mykalegebirges gelegenen Stadt Melie beteiligt. Durch glückliche Kämpfe mit dem südöstlich benachbarten Magnesia am Mäander konnte die Stellung gefestigt werden. Als ein unausweichliches Schicksal der Griechenstädte an der ägäischen Ostküste erschien es, früher oder später in die Gewalt der barbarischen Reiche östlich des schmalen, hellenisch besiedelten Meeresrandes zu geraten. Für Ephesos war es schließlich in der Mitte des 6. Jahrhunderts v. Chr. soweit, als der ebenso mächtige wie durch seinen sagenhaften Reichtum berühmte König Kroisos (der Anführer des nach dem Besitz der Küste strebenden Lyderreiches) die Stadt belagerte. Er zwang die Stadt zwar zur Anerkennung seiner Oberhoheit, aber im übrigen soll er sie gut behandelt haben.[187]
Nach einem demokratischen Zwischenspiel um das Jahr 550 v. Chr. fiel Ephesos 546/545 v. Chr. an das junge persische Großreich unter Kyros dem Großen. In den folgenden Jahrzehnten schlängelte sich Ephesos geschickt durch die Wirren der Zeit. Während des Ionischen Aufstandes (499-494 v. Chr.), bei welchem sich die griechischen Städte Kleinasiens gegen die Oberhoheit der Perser auflehnten, blieb Ephesos neutral.[188]
Im peloponnesischen Krieg (431-406 v. Chr.), in welchem es zwischen Sparta (Peloponnesischer Bund) und Athen (Delisch-Attischer Seebund) um die Vormacht im griechischen Raum ging und welcher die griechischen Städte schwächte, zeichneten sich die ersten Probleme Athens und seiner Verbündeten ab. Als Vorteile zugunsten Sparta sichtbar wurden, traten die Ephesier auf deren Seite ein und halfen ihren neuen Waffengefährten in der letzten Schlacht dieses Krieges bei Aigospotamoi bei der Vernichtung der Flotte, auf der die Macht Athens beruhte.[189]
In der Zeit danach spielte Ephesos als wichtiger Anker- und Waffenplatz der Spartaner im Kampf gegen Persien eine Zeit lang eine wichtige Rolle – kehrte jedoch im sogenannten Frieden des Antalkidas (386 v. Chr.), der aber in Wirklichkeit ein Diktat des Perserkönigs war, wieder unter persische Oberhoheit zurück. Diese endete erst mit dem Sieg des makedonischen Königs Alexander des Großen am Granikos 334 v. Chr. Auf seinem Siegeszug kam dieser auch nach Ephesos, das von dem Rhodier Memnon nochmals

[187] vgl. Scherrer 1995, S. 15-16.
[188] vgl. Letzner 2010, S. 11.
[189] vgl. Scherrer 1995, S. 17.

kurzfristig für den Perserkönig zurückgewonnen worden war, und stellte die Demokratie wieder her.[190]

Dazwischen ereignete sich in Ephesos der Brand des Artemisions. Es wird angenommen, dass Herostat im Jahr 356 v. Chr. den Tempel in Brand gesetzt hat. Das Motiv wird entweder in der Geltungssucht oder im Wahnsinn des Brandstifters vermutet. Der Wiederaufbau des Tempels zog sich derartig in die Länge, sodass noch Alexander der Große während seines Siegeszuges im Jahr 334 v. Chr. anbot, den Neubau zu unterstützen. Dieses Angebot wurde jedoch abgelehnt, da die Ephesier befürchteten, durch die Annahme der Hilfe dem makedonischen König eine Einflussnahme auf Stadt und Heiligtum genehmigen zu müssen. Politisch gesehen konnte sich nochmals kurzfristig ein Tyrann namens Hegesias etablieren, der aber im Jahr 323 v. Chr. von seinen Mitbürgern ermordet wurde.[191]

Hellenismus

Wichtiger als die Tyrannis war der Umstand, dass nach Alexanders Tod dessen riesiges Reich zerbrach. Seine unmittelbaren Nachfolger, die Diadochen, die jeweils ihre eigenen machtpolitischen Interessen vertraten, führten daraufhin um das Erbe in den folgenden Jahren heftige militärische Auseinandersetzungen miteinander. Ephesos gehörte zu den Leidtragenden dieser Kriege, aber zugleich auch zu den Gewinnern. Die gravierendste Veränderung wurde dabei von Lysimachos (König 305-281 v. Chr.) herbeigeführt: Nach dem Vorbild Alexanders, der sich als Städtegründer ausgezeichnet hatte, erzwang Lysimachos die Neugründung und Verlegung von Ephesos an ihren heutigen Standort und benannte die Stadt um in Arsinoeia (dem Namen seiner Frau Arsinoë folgend). Diese Umbenennung hatte aber keinen langen Bestand, da nach Lysimachos Tod im Jahr 281 v. Chr. die Stadt an die Seleukiden fiel.[192] Die von ihm gegründete Stadt wurde zu (Neu-)Ephesos, neben dem stets weiterbestandenen Alt-Ephesos rund um das Artemision.[193]

246 v. Chr. starb der seleukidische König Antiochos II. Theos in Ephesos und bald danach geriet die Stadt unter die Herrschaft der ägyptischen Ptolemäer. Sie blieb in dieser, bis der Seleukide Antiochos III. die ptolemäischen Außenbesitzungen in Kleinasien eroberte.[194] Nachdem Antiochos III. 195 v. Chr. den Erzfeind Roms, den karthagischen Feldherrn Hannibal, in Ephesos empfangen hatte, eskalierte schließlich die Lage zu einem Krieg (192-

[190] vgl. Scherrer 1995, S. 17.
[191] vgl. Letzner 2010, S. 11.
[192] vgl. Letzner 2010, S. 12.
[193] vgl. Scherrer 1995, S. 20.
[194] vgl. Scherrer 1995, S. 20-21.

188 v. Chr.).[195] 190 v. Chr. unterlagen die Seleukiden dann den Römern in der Schlacht von Magnesia am Sipylos. Daraufhin mussten sie 188 v. Chr. ganz Kleinasien bis zum Taurus räumen. Das Königreich Pergamon, welches die Römer unterstützt hatte, bekam seinen Lohn durch den Zugewinn großer Gebiete Westkleinasiens – und damit auch Ephesos. Als schließlich der letzte pergamenische König, Attalos III., kinderlos 133 v. Chr. starb, vermachte er sein Reich testamentarisch „dem römischen Volk". So wurde Ephesos, ebenso wie Pergamon, Teil der neu eingerichteten Provinz Asia.[196]

Nach diesen Kriegswirren war es nicht verwunderlich, dass der pontische König Mithridates Eupator Dionysos, nachdem er sein Reich bis zur Krim ausgedehnt hatte, auch nach Süden und Westen expandierte. So fiel er 89 v. Chr. in die Provinz Asia ein und begann damit einen Siegeszug bis nach Griechenland, wobei ihn anfänglich viele Griechen als Befreier freudig willkommen hießen – so auch Ephesos, wo man zunächst nur die Bildnisse von Römern umstürzte.[197]

In unmittelbarem Zusammenhang mit Mithridates zu Beginn des ersten Mithridatischen Krieges (89-85 v. Chr.) steht eine der dunkelsten Episoden in der Geschichte der Stadt. So befahl nämlich der pontische König 88 v. Chr. zu einem bestimmten Stichtag die Ermordung aller Römer und Italiker in der Provinz. Dieser Befehl ist in die Geschichtsschreibung als „Ephesische Vesper" eingegangen – etwa 80.000 Menschen wurden dabei getötet. Für diesen Befehl hatte Mithridates zwei Gründe: Zum einen musste er den Krieg gegen Rom finanzieren und konnte dafür nicht die gerade von ihm Befreiten zur Kasse bitten. Zum anderen konnte er davon ausgehen, dass Städte, die sich an dem Massaker beteiligt hatten, aus Furcht vor römischer Rache nicht, wie gewohnt, die Seiten wechselten, wie sie wollten. Ephesos musste dafür wenige Jahre später, da selbst die Tempelasylie keinen Schutz mehr geboten hatte, im Jahr 84 v. Chr. – nach dem Sieg von L. Cornelius Sulla über Mithridates – schwer büßen. Dann verlor die Stadt Teile ihres Territoriums, die Häupter der antirömischen Partei wurden hingerichtet, es erfolgte eine Brandschatzung und weiters mussten auch die ausgefallenen Steuern der Jahre 88-85 v. Chr. nachgezahlt werden.[198]

Die Neuordnung des Augustus

48 v. Chr. kam Gaius Iulius Caesar nach Ephesos und gewährte den Provinzialen großzügige Erleichterungen in der Steueraufbringung. Weniger großzügig war jedoch der nächste, nach Ephesos gekommene Römer – nämlich Marcus Antonius, der 41 v. Chr. diese Erleich-

[195] vgl. Letzner 2010, S. 12.
[196] vgl. Scherrer 1995, S. 20-21.
[197] vgl. Karwiese 1995, S. 72.
[198] vgl. Letzner 2010, S. 13.

terungen nicht nur zurücknahm, sondern sogar kaum erfüllbare Forderungen stellte. 33 v. Chr. ist Antonius nach Ephesos zurückgekehrt, um hier mit der ägyptischen Königin Kleopatra zu überwintern und den unausweichlich gewordenen Krieg mit seinem Erzfeind Octavian (dem späteren Kaiser Augustus) um die Alleinherrschaft über das römische Reich vorzubereiten. Die Siege Octavians in der Seeschlacht von Aktium 31 v. Chr. und in der Landschlacht des folgenden Jahres vor den Toren Alexandrias beendeten diese lange Zeit blutiger römischer Bürgerkriege. Die Republik, welche sich unfähig gezeigt hatte, ein Weltreich zu regieren und ordentlich zu verwalten, war tot und an ihre Stelle trat eine neue, für die nächsten 300 Jahre bestehende Staatsform: der von Octavian-Augustus begründete Prinzipatsstaat. Weiters wurden die völlig zerrütteten Verhältnisse in Asia neu geordnet und der Sitz des römischen Statthalters von Pergamon nach Ephesos verlegt.[199]

Ephesos als Weltstadt der römischen Kaiserzeit

Die Entwicklung der Stadt während der Kaiserzeit konnte auch nicht durch zahlreiche Erdbeben eingeschränkt werden, welche die Region erschütterten. Um die Mitte des 1. Jhs. n. Chr. war Ephesos dann bereits die zweitgrößte Stadt des Orients, wie Seneca feststellte, und für das 2. Jh. wurde die ephesische Bevölkerung auf ca. 300.000 Menschen geschätzt. Damit bewegte sich die Stadt auf einem ähnlichen Niveau wie Alexandria, für das in etwa die gleiche Einwohnerzahl angenommen wird.[200] Einer Weltstadt entsprechend vielschichtig und bunt war die multikulturelle und multinationale Palette der Bevölkerung: Sie reichte von altetablierten, mehr oder weniger romanisierten Eingesessenen über italische, im Laufe der Zeit mehr oder weniger hellenisierte Zuzügler bis zu einer stattlichen Judengemeinde, Ägyptern (Alexandrinern), Rhodiern und anderen, die in Ephesos Handelsfaktoreien unterhielten.[201]

Der Aufschwung der Stadt beruhte in erster Linie auf ihrem Hafen als Umschlagplatz für das, was aus dem ferneren Orient über Ephesos exportiert wurde und für das, was die reiche Provinz selbst produzierte oder aus dem Westen des Reiches importierte. Das Artemision wurde ein mächtiger Wirtschaftskörper, welches als eines der sieben Weltwunder von Touristen aus aller Welt besucht wurde, weithin bekannt war als „Kreditbank von Asia“ und zugleich auch ein wichtiger Arbeitgeber war.[202]

Katastrophen prägten die Zeit in Ephesos ab der Mitte des 2. Jahrhunderts. So brach beispielsweise während eines Feldzuges von Lucius Verus (161-169 Mitkaiser von Marcus

[199] vgl. Scherrer 1995, S. 22.
[200] vgl. Letzner 2010, S. 13-14.
[201] vgl. Scherrer 1995, S. 28.
[202] vgl. Scherrer 1995, S. 26.

Aurelius), welcher von Ephesos aus im Jahr 162/163 gegen die Parther aufgebrochen war, eine Epidemie aus. Diese wurde von den rückkehrenden Truppen nach Ephesos und die übrigen Regionen des Reichs eingeschleppt und hatte verheerende wirtschaftliche Konsequenzen. Dies wiederum war eine Ursache dafür, dass die Zentralmacht in der Folgezeit nicht mehr in der Lage war, nach Erdbebenschäden zu helfen. Weiters kam erschwerend hinzu, dass auch der Hafen zunehmend verlandete – so hatte bereits Hadrian einen Kanal anlegen lassen, um den Hafen weiterhin offen zu halten. [203]

Umbruch zur Spätantike

Erst in der Ära der Soldatenkaiser (235-284) ist es deutlich abwärts gegangen. So wurden nämlich jeweils die Kaiser von immer barbarischer werdenden Soldaten nach spätestens acht Jahren umgebracht. In dieser Zeit der Unruhen suchte auch eine Horde seefahrender Goten, die aus dem Schwarzmeergebiet aufgebrochen waren, Ephesos heim. Die Verteidigungsfähigkeit der Stadt war gering, da sie 262 durch ein schweres Erdbeben geschwächt worden war. So gestaltete es sich für die Goten nicht schwierig, das reiche Artemision mit seinen gewaltigen, im Laufe vieler Jahrhunderte angesammelten Schätzen zu plündern und Feuer zu legen. Die Schäden konnten nach dem Abzug der Barbaren behoben werden – der alte Glanz war aber vorbei. Artemis hatte es nicht geschafft, ihr eigenes Haus zu schützen. Daher drängte sich wohl folgende unangenehme Frage auf: Wie konnte sie angesichts dieser Niederlage noch länger die Beschützerin der Stadt sein?[204]

Die Mitte dieses turbulenten 3. Jahrhunderts hat den Beginn von Christenverfolgungen in einem bisher nicht gekannten Ausmaß gesehen. In dieser Zeit entstand auch die Sieben Schläfer-Legende, wonach sieben fromme Jünglinge in den Höhlen an der Nordostseite des Panayır Dağ nahe dem Meter-Heiligtum in der Zeit der Bedrängnis durch Kaiser Decius in Schlaf gefallen seien und unter dem christlichen Kaiser Theodosius II. unversehrt wiederauferstanden seien. Es kam in der Folgezeit zu einer immer schneller werdenden Abwärtsentwicklung des Reiches, welche ihr Ende erst mit der Machtergreifung Diokletians 284 fand. Mit seiner Herrschaft beginnt der spätrömische Dominatsstaat. Diokletian startete zwar nochmals eine reichsweite Christenverfolgung, brach diese jedoch 303 offenbar aus Gründen der Staatsräson ab. Nach der Wiederherstellung der Reichseinheit hat Diokletian durch ein neues, striktes Steuererhebungssystem, sowie durch die Festlegung von

[203] vgl. Letzner 2010, S. 14.
[204] vgl. Scherrer 1995, S. 30.

Höchstpreisen für alle Waren und Dienstleistungen zur Bekämpfung der Inflation, das Vertrauen in die Reichswährung wiederherstellen können.[205]

Sieg des Christentums

Nach heftigen Wirren und Turbulenzen trat erst 324 mit der Erringung der Alleinherrschaft durch Constantin I. wieder dynastische Stabilität und Reichseinheit ein. Unter Constantin I. ist das bisher geächtete Christentum zur erlaubten Religion geworden. Unter dem seit 350 als Alleinherrscher regierenden, von den drei Söhnen Constantins letztlich übriggebliebenen und überaus christlich-bigotten Constantinus II. ist auch Ephesos eine christliche Stadt geworden. Mit dem Edikt des Kaisers Theodosius I. im Jahr 380 wurde das Christentum schließlich zur Staatsreligion erhoben und das Heidentum war damit nun auch „amtlich" gestorben. Das Artemision ist wohl schon bald nach dem Sieg des Christentums als Steinbruch ausgebeutet worden, was man daran erkennen kann, dass die Überreste des Tempels verglichen mit seiner einstigen Größe sehr gering sind.[206]

431 wurde auf dem dritten Ökumenischen Konzil ein für das Christentum immer noch geltender Beschluss gefasst: So wurde nämlich Maria zur Gottesgebärerin (Theotokos) erhoben. Daraus entwickelte sich aber so ein heftiger theologischer Streit, sodass sich Theodosius II. (Kaiser 408-450) gezwungen sah, die führenden Köpfe der miteinander streitenden Parteien abzusetzen. 449 fand noch ein weiteres Konzil statt, welches noch chaotischer ablief und als sogenannte „Räubersynode" in die Kirchengeschichte einging und nicht zu den offiziellen Konzilien gerechnet wird.[207]

Das Leben im Gebiet der antiken Stadt sollte sich ab Mitte des 6. Jahrhunderts grundlegend ändern. Den Anfang machte Erzbischof Johannes III., der Teile der Verwaltung in die Johannesbasilika, welche unter Justinian (Kaiser 527-565) errichtet worden war, verlegte. Von 610 bis 654/655 erlebte die Stadt noch eine Nachblüte, da Ephesos zur Hauptstadt des byzantinischen Militärbezirkes Thrakesion wurde.[208]

Arabische Bedrohung und Untergang von Ephesos

Auf der arabischen Halbinsel jenseits des hellenistisch-römischen Kulturbereiches war nach dem Judentum und Christentum der Islam als eine neue Religion entstanden. Diese trat von Anfang an mit dem Anspruch auf, eine Weltreligion zu sein und empfand es als ihren Auftrag, alle Menschen der Lehre seines Begründers Mohammed zuzuführen. Bereits im Jahr 688 plünderte der omaijadische Kalif Muawija mit einer arabischen Flotte Ephesos bei der

[205] vgl. Scherrer 1995, S. 30-31.
[206] vgl. Scherrer 1995, S. 32.
[207] vgl. Letzner 2010, S. 14.
[208] vgl. Letzner 2010, S. 15.

Rückkehr von einer erfolglosen Expedition gegen Konstantinopel – dasselbe tat auch der arabische Admiral Maslama 716/717. Da das ganze, einst von Lysimachos abgesteckte Stadtgebiet nicht zu halten und nur noch spärlich bewohnt war, reduzierte man es auf jenes Ausmaß, das von der byzantinischen Stadtmauer umschlossen wird. Das Schicksal des hellenistisch-römischen Ephesos, das im verkleinerten Maßstab noch etwa bis in das 10. Jahrhundert existiert hat, erfüllte sich schließlich, als der Hafen infolge des Zurückweichens der Küste nach Westen nur noch durch einen schlauchartigen Kanal mit dem Meer verbunden war, endgültig verlandete. Im 11. Jahrhundert sollte das alte Ephesos schließlich endgültig untergehen – so wurde aus der Hafenstadt Ephesos eine Binnenstadt auf dem Ayasolukberg.[209]

Bereits seit dem 6. Jahrhundert war rund um die Johanneskirche eine Siedlung entstanden, welche in gesünderer Luft lag als das in zunehmendem Maß an einer Fiebersumpfebene gelegene Ephesos. Ein weiterer Vorteil war der, aus den Steinen und Bauteilen des Ruinenfeldes der hellenistisch-römischen Stadt und des Artemisions errichtete, Mauerring, welcher Schutz gegen Seepiraten und feindliche Flotten bot. Dieses Ephesos wurde später nach dem Türken Hagios Theologos (Heiliger Theologe) „Ayasoluk“ genannt. Die Rolle von Ephesos als Hafenstadt ging auf Scalanuova über.[210]

Das christliche Ephesos sollte nur eine, wenn auch langandauernde, Episode bleiben. Um 1300 geriet Ayasoluk-Altoluogo, ebenso wie die gesamte Region, in die Gewalt der Aydınoğlu – einer Fürstendynastie des türkischen Stammes der Seldschuken. Unter einem ihrer Söhne hat Ayasoluk-Altoluogo eine letzte Hochblüte erlebt, von der außer der, nach ihrem berühmten Baumeister Ali von Damaskus benannten, Moschee auch Bäder und Mausoleen im Stadtgebiet des heutigen Selçuk Zeugnis ablegen.[211]

Am Beginn des 15. Jahrhunderts endete die Herrschaft der Aydınoğlu mit der Machtergreifung des Hauses Osman. Der Weltreichsanspruch der Dynastie konnte nicht eine provinzielle Residenz in Ayasoluk sein, sondern zielte darauf ab, sich der Hauptstadt des byzantinischen Reiches zu bemächtigen – der noch immer glanzvollen Kaiserstadt am Bosporus, auf die sich dieses Reich schließlich reduziert hatte und welches 1453 in die Hände des Sultans Fatih Mehmet fallen sollte.[212]

[209] vgl. Scherrer 1995, S. 34-35.
[210] vgl. Scherrer 1995, S. 35.
[211] vgl. Scherrer 1995, S. 35.
[212] vgl. Scherrer 1995, S. 35-36.

War das verödete ephesische Ruinenfeld schon längst zu einem Friedhof der Geschichte geworden, so sank Ayasoluk nun zu einer bedeutungslosen Provinzsiedlung des großen osmanischen Reiches herab.[213]

6.1.4 Baudenkmäler und Funde

Neben dem wohl berühmtesten Baudenkmal Ephesos', dem Tempel der Artemis, gibt es noch viele weitere bemerkenswerte Baudenkmäler und Funde. Eine Auswahl dieser – allen voran das Artemision – soll nun im Folgenden erörtert werden.

Artemision

Das Heiligtum der Artemis von Ephesos, als eines der Sieben Weltwunder der Antike, war für die Geschichte der Stadt Ephesos zweifellos der wichtigste Bau. Der Ort, an dem der Tempel der Artemis errichtet wurde, war aufgrund eines sumpfigen Geländes denkbar ungeeignet und musste mit Hilfe von Holzkohle, welche unter dem Tempel in den Sumpf eingebracht wurde, getrocknet und stabilisiert werden.[214] Auch heutzutage kann man am Gelände kaum noch etwas sehen (siehe Abb. 157), und das Wenige, das sichtbar ist, liegt je nach Jahreszeit bzw. nach heftigen Niederschlägen unter Wasser.[215]

Artemiskult

Wie kam es dazu, dass gerade die griechische Göttin Artemis in Ephesos angebetet wurde? Angeblich wurde die Anbetung durch ein Bildnis der Göttin, das vom Himmel gefallen sein soll, ausgelöst. Artemis, Tochter des Zeus, gehörte zu den zwölf höchsten Gottheiten in der griechischen Mythologie und war die Göttin der Jagd, des Mondes und Hüterin von Frauen und Kinder. Man fand drei Statuen in den Ruinen des Artemisions, auf welchen sie auf ihrem Bauch mehrfache, nicht einfach zu identifizierbare Rundungen hat (siehe Abb. 160). Da keine Überlieferungen existieren, gibt es verschiedene Spekulationen, welche von Stierhoden, über Eier bis hin zu Früchten, reichen, aber alle zu demselben Ergebnis führen, nämlich dass Artemis in Ephesos auch als Fruchtbarkeitsgöttin verehrt wurde.[216]

---

[213] vgl. Scherrer 1995, S. 36.
[214] vgl. URL: http://www.weltwunder-online.de/antike/tempel-artemis-ephesus.htm, aufgerufen am 29.11.2015.
[215] vgl. Letzner 2010, S. 17.
[216] vgl. URL: http://www.weltwunder-online.de/antike/tempel-artemis-ephesus.htm, aufgerufen am 29.11.2015.

Abbildung 160 Statue der Artemis

Die zwei Artemis Tempel

Spricht man vom Tempel der Artemis in Ephesos als Weltwunder, so sind eigentlich zwei Tempel damit gemeint. Als der erste durch Brandstiftung zerstört worden war, baute man sofort einen zweiten Tempel an der gleichen Stelle, welcher an Pracht und Größe mit dem ersten vergleichbar war. Somit kann man sagen, dass beide Tempel Meilensteine der antiken Kunst und Architektur und wahrscheinlich die schönsten Bauwerke im griechisch-römischen Altertum waren.[217]

Der erste Tempel der Artemis wurde um 460 v. Chr. vollendet und seine Bauzeit betrug etwa 100 Jahre. Nach einer Inschrift, welche gefunden wurde, trug der reiche König Kroisos von Lydien einen Teil der Kosten für den Tempelbau. Er hatte kurz zuvor Ephesos erobert und versuchte auf diese Weise womöglich die Zuneigung der Ephesier zu erkaufen.

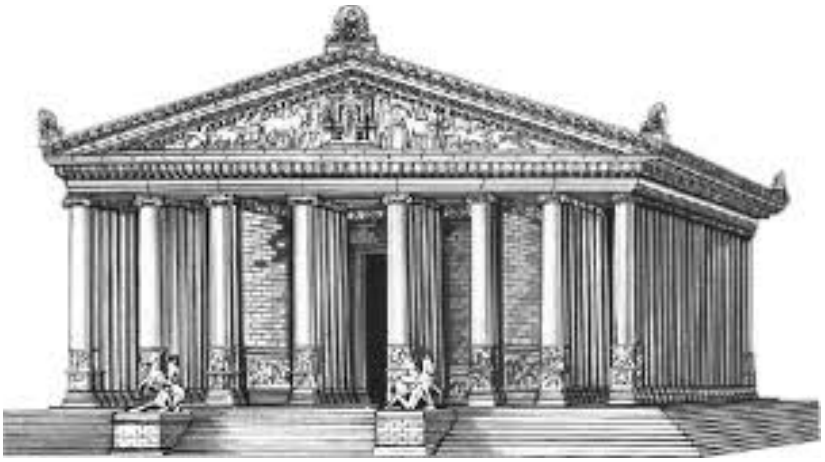


Abbildung 161 Rekonstruktion des ersten Artemis Tempels

[217] vgl. URL: http://www.weltwunder-online.de/antike/tempel-artemis-ephesus.htm, aufgerufen am 29.11.2015.

Bereits kurz nach dem Brand und der Zerstörung des ersten Artemis Tempels wurde (wie zuvor erwähnt) an gleicher Stelle (da dieser Platz sozusagen „von den Göttern vorherbestimmt“ war) ein zweiter, nur wenig größerer errichtet, welcher in die Liste der Weltwunder von Antipatros aufgenommen wurde. Das zweite Artemision wurde ebenfalls vollständig aus Marmor gebaut, wies aber weniger Verzierungen und Friese auf als der erste Tempel. Die Zerstörung des zweiten Tempels erfolgte schließlich 262 n. Chr. durch die Goten (siehe „6.1.3 Geschichtlicher Überblick“). Als das römische Reich 100 Jahre später christlich wurde, ging Ephesos und mit ihm der Kult um die Göttin Artemis unter und die Ruine des Artemisions diente künftig als Steinbruch für Neubauten in der näheren Umgebung (siehe ebenfalls „6.1.3 Geschichtlicher Überblick“).[218]

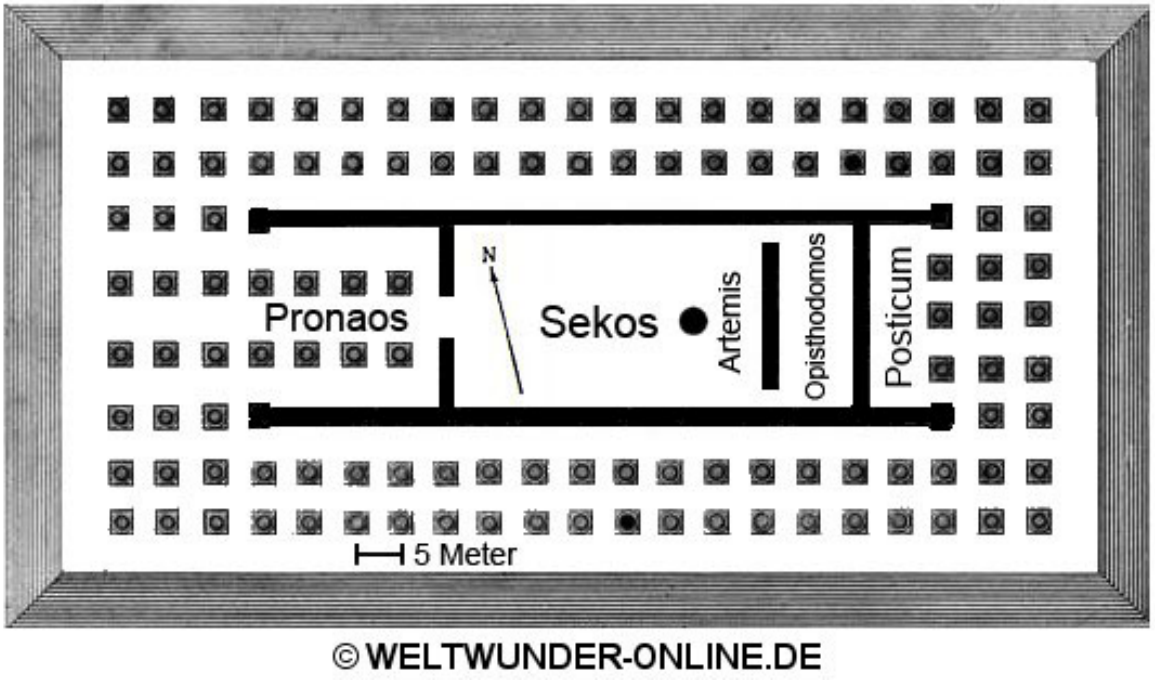


Abbildung 162 Rekonstruktion des Artemisions

Forschungsgeschichte

Das Artemision wurde Ende 1869 von J. T. Wood nach siebenjähriger Suche entdeckt und ausgegraben. Sein Nachfolger – der Engländer D. G. Hogarth – grub 1904/1905 ebenfalls für das Britische Museum aus und erforschte nicht nur den Bau, sondern auch ältere Anlagen in der sogenannten Zentralbasis im Inneren des Tempelhofs. Im Jahr 1965 wurden die Ausgrabungen vom Österreichischen Archäologischen Institut wiederaufgenommen und bis heute weitergeführt. Die Ergebnisse dieser Ausgrabungen rückten nicht nur den Tempel und den großen Hofaltar in den Mittelpunkt des Interesses, sondern auch die Zentralbasis im

[218] vgl. URL: http://www.weltwunder-online.de/antike/tempel-artemis-ephesus.htm, aufgerufen am 29.11.2015.

Inneren der Tempelanlage konnte noch einmal erforscht werden. Als ein sensationelles Ergebnis wurde die Existenz eines Ringhallentempels aus dem 8. Jh. v. Chr., des bis heute ältesten in der griechischen Architektur, angesehen. In Zusammenhang mit diesem Bau wurde auch ein bedeutender Hortfund entdeckt, hauptsächlich aus Bernstein, der vermutlich zum Brustschmuck eines Kultbildes gehört haben wird.[219]

Der Stadtentwurf

Grundvoraussetzung für das Gedeihen einer Stadt ist die Schaffung einer ausreichenden Infrastruktur – dazu zählen unterschiedliche Aspekte, wie z.B. der Stadtentwurf, die Verteidigungsanlagen oder die Wasserversorgung, welche eine Stadt lebenswert machen.[220]

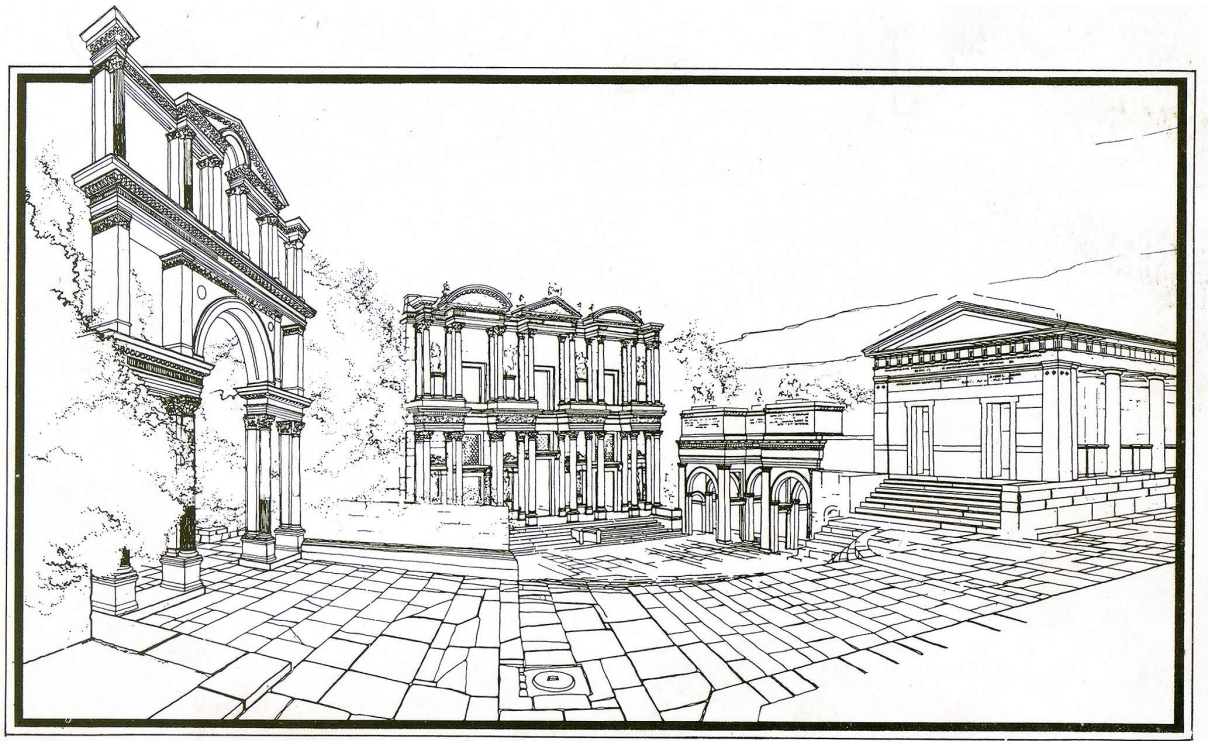


Abbildung 163 Rekonstruktion des Platzes vor der Celsus-Bibliothek

Das heutige Stadtbild von Ephesos, das nur von einigen Straßenzügen und Plätzen bestimmt wird, entspricht nicht der antiken Realität: Als Lysimachos nämlich die Stadt neu anlegen ließ, wurde eine Stadt mit orthogonalem Straßensystem konzipiert, wie es in der hellenistischen Welt dieser Zeit üblich war. Dieses Schachbrett-System wurde nur durch den Embolos (Kuretenstraße) unterbrochen. Im Rahmen des Rasters konnten dann Wohnhäuser, Tempel oder öffentliche Bauten errichtet werden. An einigen Stellen lässt sich dieses Straßensystem auch gut erkennen – so nehmen z.B. die Hanghäuser am unteren Embolos jeweils einen Wohnblock ein. Aber auch die Topographie beeinflusste die Gestaltung der

[219] vgl. Scherrer 1995, S. 46.
[220] vgl. Letzner 2010, S. 25.

Straßen, denn so wurden, um die Höhenunterschiede zu überbrücken, Treppenanlagen eingeführt (die sogenannten „Stiegengassen“).[221]

Die Celsus-Bibliothek

Einen architektonischen Höhepunkt im Stadtbild von Ephesos bildet die Celsus-Bibliothek, siehe Abb. 164.

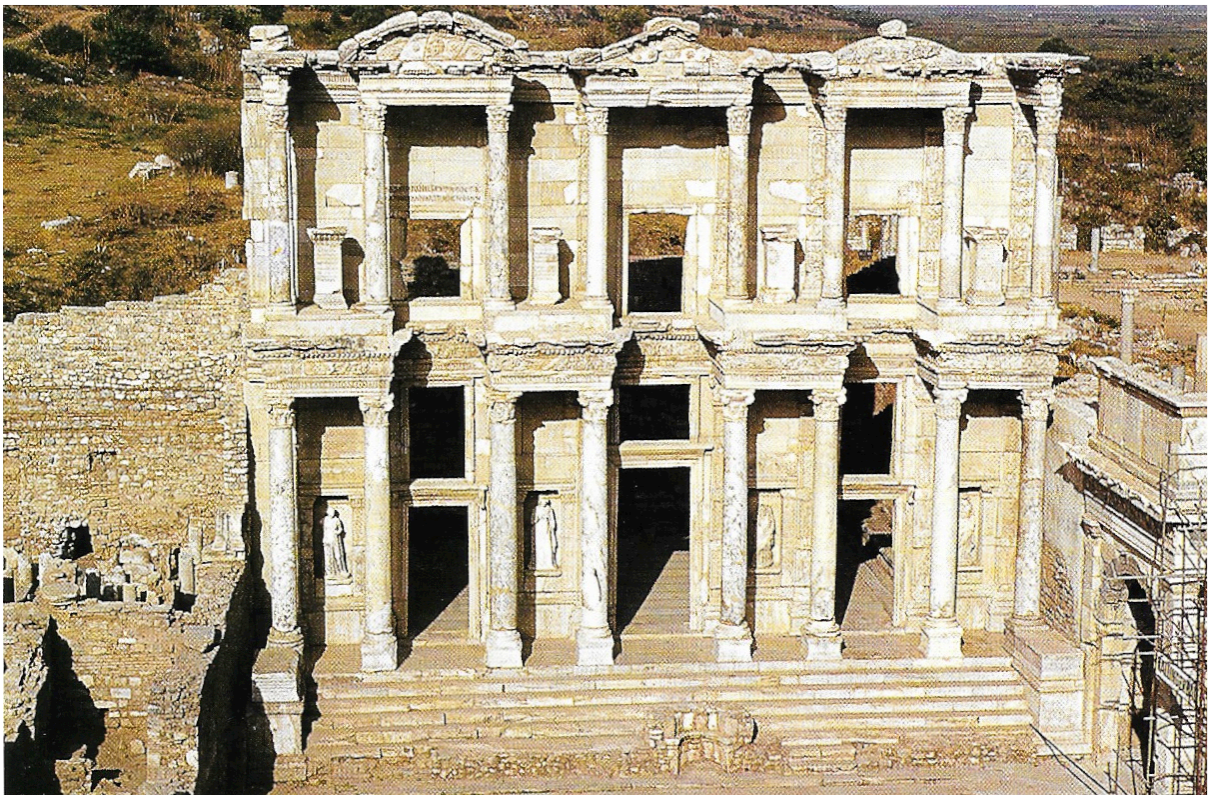


Abbildung 164 Celsus-Bibliothek – Vorderansicht

Dies liegt vor allem am Wiederaufbau der Fassade mit weitgehend originalen Bauteilen. Der aufwändige Bau wurde von Tiberius Iulius Celsus Polemaeanus gestiftet, aber die weitgehend erhaltene Bauinschrift nennt C. Iulius Aquila (seinen Sohn) als Ausführenden. Die Celsus-Bibliothek war nicht nur eine Bildungsstätte, sondern zugleich auch Grabmal, denn so verstand Celsus die Bibliothek als repräsentative Umgebung für sein Grab, das unter dem Gebäude liegt. Vom Innenraum der Bibliothek aus konnte man schon damals durch zwei Öffnungen einen Blick auf den Sarkophag werfen.[222]

In den vier Nischen des Untergeschoßes befanden sich die Tugenden des Celsus – nun nur mehr Abgüsse, da die Originale im Wiener Ephesos-Museum ausgestellt sind. Es folgen von links nach rechts: Sophia (Weisheit), Arete (Charakter), Ennoia (Urteilskraft) und Episteme (Sachverstand) – die typischen Erwartungen an einen hohen römischen Beamten. Der

[221] vgl. Letzner 2010, S. 25-26.
[222] vgl. Letzner 2010, S. 81-82.

eigentliche Bibliotheksbestand, nämlich die Buchrollen, war in schrankartigen Nischen untergebracht, die in den beiden Obergeschoßen des Raumes über Galerien zugänglich gewesen sind.[223]

Die Celsus-Bibliothek wurde im Jahr 262 n. Chr. durch einen Brand weitgehend zerstört – erhalten blieb jedoch die zweigeschoßige Fassade. Im 4. oder 5. Jh. n. Chr. wurde die Bibliotheksfassade dann in ein Nymphaeum (Heiligtum) umgebaut.[224]

Das Tor des Mazaeus und Mithridates

Direkt neben der Celsus-Bibliothek fällt besonders das sogenannte Tor des Mazaeus und Mithridates auf, siehe Abb. 165.

Abbildung 165 Das Südtor der Agora – das Tor des Mazaeus und Mithridates

Der Torbau hat drei von Bögen überspannte Durchgänge, welche in der Tiefe unterteilt sind. Die Südseite als Hauptansichtsseite ist im Mittelteil U-förmig eingezogen. Die Durchgänge wiederum sind untereinander durch verzierte Türgewände verbunden und in die Außenmauern sind je zwei halbrunde Nischen eingebaut. Die Bauinschrift (teils auf Griechisch, teils auf Latein) besagt, dass Mazaeus und Mithridates, Freigelassene des Kaiserhauses, 3 v. Chr. das Tor zu Ehren des Kaisers Augustus, seiner Gattin Livia, seines Schwiegersohnes Agrippa und seiner Tochter Julia errichten ließen. Viele weitere Inschriften

[223] vgl. Scherrer 1995, S. 132-134.
[224] vgl. Letzner 2010, S. 86.

lassen sich am gesamten Bau finden, welche sich entweder auf Baumaßnahmen der Umgebung beziehen oder Preisedikte und andere behördliche Verordnungen wiedergeben.[225] Dieses Denkmal wurde weitgehend mit Originalmaterial zwischen 1979 und 1988 wieder aufgerichtet. Es lassen sich bei dem Torbau zwei Bauphasen unterscheiden. Zur ersten gehörten zwei Flügelbauten, die zweigeschoßig angelegt waren. Dabei handelte es sich vermutlich um die Gräber der Stifter. Als die Tetragonos-Agora (diese Anlage diente als Handelsmarkt[226]) im 1. Jh. n. Chr. umgebaut wurde, musste der östliche Anbau, der des Mithridates, verkleinert werden – während der westliche, der dem Mazaeus zugeschrieben wurde, spätestens dann abgerissen wurde, als die Celsus-Bibliothek entstand.[227]

Das Theater

Zu den beeindruckendsten Bauten in Ephesos zählt neben dem Artemision auch das große Theater am Hang des Panayır Dağ (siehe Abb. 166).

Abbildung 166 Das große Theater

Um heutzutage nachvollziehen zu können, welche Bedeutung solch ein riesiges Theater damals für die Bewohner dieser Metropole gehabt hat, muss man zunächst mitbedenken, dass Theateraufführungen im Kulturleben in der antiken Welt im Sinn der Massenunterhaltung

[225] vgl. Scherrer 1995, S. 140.
[226] vgl. Letzner 2010, S. 89.
[227] vgl. Letzner 2010, S. 87.

eine wichtige Rolle spielten und dass schon alleine aus diesem Grund kaum eine Stadt ohne entsprechenden Bau auskommen konnte.[228]

Der früheste Beleg für ein Theater an dieser Stelle deutet auf 100 v. Chr. hin und steht im Kontext eines Bauprogramms, das seine Ursache in der Einrichtung der Provinz Asia (siehe „6.1.3 Geschichtlicher Überblick“) hatte. Gegenüber dem heutigen Zustand war dieser Bau aber eher bescheiden. In der ersten Bauphase gab es ein kleines Bühnengebäude, die Orchestra (in ihrer ursprünglichen Funktion ein Tanzplatz des Chores), einen Entwässerungskanal und einen Zuschauerraum (vermutlich nur ein erster Rang). Die städtische Entwicklung im 1. Jh. n. Chr. machte dann eine Erweiterung des Theaterbaus notwendig – so wurden unter den Kaisern Nero und Domitian in den Jahren 87 und 92 diverse Veränderungen (vor allem im Bühnenbereich und im Bereich des Zuschauerraums) durchgeführt. Ein weiterer Ausbau des Zuschauerraumes mit einem dritten Rang erfolgte zu einem nicht näher bestimmbaren Zeitpunkt, definitiv aber vor dem Erdbeben von 262. Mit dieser abschließenden Erweiterung des Zuschauerraums fanden nun rund 25.000 Menschen im Theater Platz. Bei weiteren baulichen Veränderungen um die Mitte des 2. Jahrhunderts ging es dann weniger um eine weitere Erhöhung der Zuschauerkapazität als vielmehr um eine Steigerung des Zuschauerkomforts (so wurden beispielsweise Sonnensegel errichtet).[229]

Beim großen Erdbeben im Jahr 262 stürzten Teile des Bauwerkes ein, was im Nord-Analemma (eine seitliche Stützmauer) dazu führte, dass die Stiegenaufgänge an dieser Stelle grundlegend verändert wurden. Eine neuerliche Zerstörung infolge eines späteren Erdbebens (zwischen 359 und 366) führte schließlich zur vollkommenen Aufgabe von dessen oberen Bereich. Aber auch schon vor diesem Ereignis hat sich die klassische Bestimmung des Theaters geändert, da es nun u.a. auch Gladiatorenkämpfe umfasste, was an diversen Graffiti an der Bühnenvorderseite zu erkennen ist.[230]

Der Hafen

Nicht zu vernachlässigen ist die Bedeutung der Hafen an der Mündung des Kaystros – so wird dieser nämlich gezielt als Dreh- und Angelpunkt in der Entwicklung der Stadt Ephesos angesehen. Auch wenn sich anfangs gute Ankermöglichkeiten boten, kam es später durch die fortschreitende Verlandung zu großen Problemen (wie bereits angesprochen). Immer wieder mussten Baumaßnahmen ergriffen werden, um den Hafen für den Schiffsverkehr offen zu halten. Dies führte schließlich bis hin zur Verlegung des Hafens (siehe Abb. 167). Das

[228] vgl. Letzner 2010, S. 94-95.
[229] vgl. Letzner 2010, S. 97-99.
[230] vgl. Scherrer 1995, S. 162.

Problem der Verlandung barg aber neben verkehrstechnischen Aspekten noch ein weitaus gravierendes, nämlich Malaria, was ein weiterer Grund dafür war, das alte Stadtgebiet aufzugeben.[231]

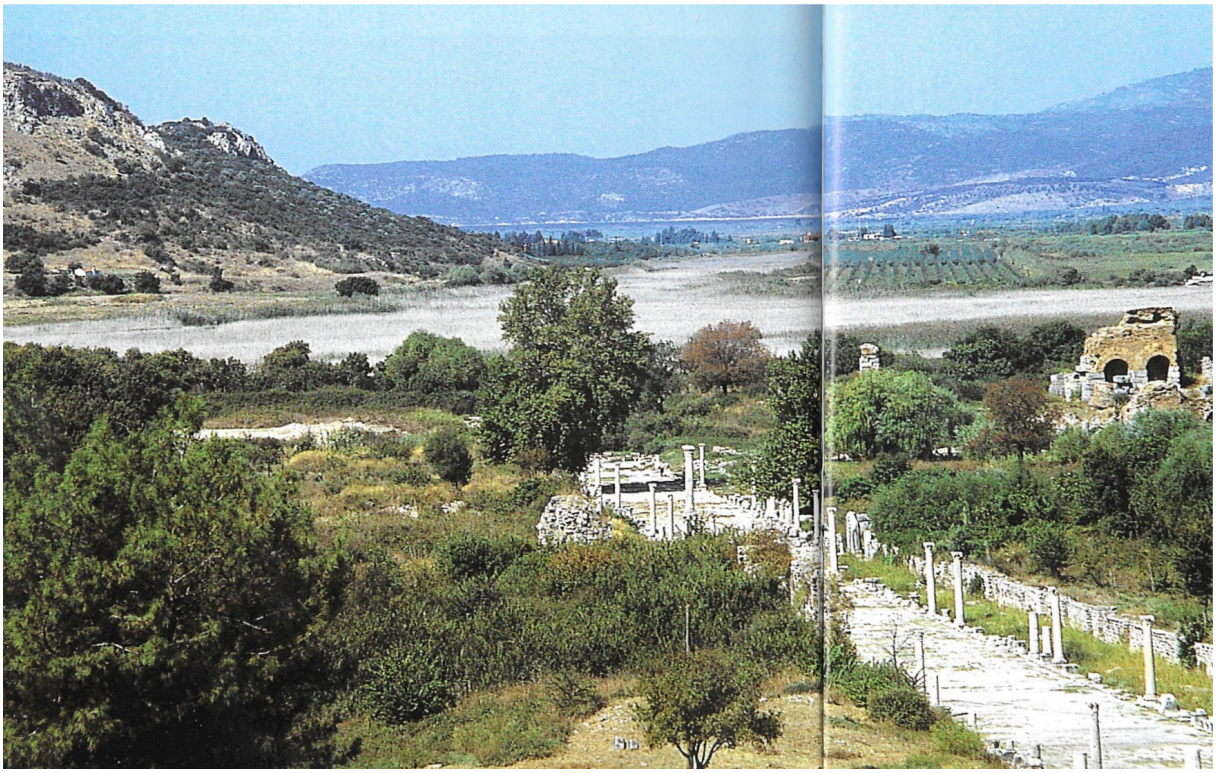


Abbildung 167 Blick auf den Hafen

Um den Hafen heute erkennen zu können, muss ein höherer Standort gewählt werden, von welchem aus dann recht gut das Hafenbecken gesehen werden kann, das in seiner jetzt wahrnehmbaren Form in trajanisch-hadrianischer Zeit (zwischen 98 und 138 n. Chr.) entstanden sein dürfte. Weiters konnten Bauten nachgewiesen werden, die für den Hafenbetrieb unabdingbar sind. Dabei handelt es sich um zwei große dreischiffige Hallen. Da in diesen Hallen keine Innenwände nachgewiesen wurden, werden diese Bauten gerne als Handelsräume interpretiert. Notwendige Speicherbauten wurde bislang nicht ausgegraben, müssten aber prinzipiell existiert haben.[232]

Nicht dem Handel dienend, aber für Repräsentationszwecke gut sind die Hafentore. Zwischen den Jahren 1896 und 1899 wurden drei Torbauten (ein südliches, ein mittleres und ein nördliches Hafentor) gefunden – sie markieren die Stellen, wo große Straßen auf das Hafengebiet treffen. Das südliche Hafentor ist inschriftlich als Stiftung eines Asiarchen um 200 n. Chr. datiert. Dies bestätigt auch die Architekturdekoration. Das mittlere Hafentor steht am Ende der Arkadiane und dürfte in hadianischer Zeit errichtet worden sein. Drei

[231] vgl. Letzner 2010, S. 101.
[232] vgl. Letzner 2010, S. 101.

Durchgänge zwischen Stützen aus je vier ionischen Säulen sind festzustellen. Das nördliche Hafentor ist nur sehr schlecht erhalten und Inschriftenreste erlauben die Deutung als Ehrenbogen für einen „Proconsul Asiae“ aus der Mitte des 3. Jahrhunderts n. Chr.[233]

Am wichtigsten in Bezug auf das Programmierprojekt sind die sogenannten Hanghäuser, da in diesen die Zahlenrätsel gefunden worden sind, welche das Programm zu lösen versucht. Daher wird diesen außergewöhnlichen Beispielen für Wohnluxus in Ephesos ein eigenes Kapitel gewidmet – nämlich das folgende: „6.2 Die Hanghäuser von Ephesos“.

6.2 Die Hanghäuser von Ephesos

Die Hanghäuser, deren Name sich von der Lage am Hang des Bülbül Dağ her erklärt, bildeten einen gewaltigen Baukomplex in Ephesos. Die Häuser, welche im Straßensystem jeweils die Fläche einer Insula (Häuserblock) einnehmen, wurden zwischen 1960 und 1983 ausgegraben und erbrachten so viele Befunde, dass deren Konservierung an Ort und Stelle notwendig wurde. So wurde dementsprechend ein Schutzbau über dem Hanghaus 2 angelegt, welches nun seit der Fertigstellung den Besuchern offensteht (siehe Abb. 168). Das Beeindruckende an den Hanghäusern ist, dass sie einen Einblick in das soziale Leben der Stadt erlauben: So kann man hier nämlich den Wandel von der herrschaftlichen Residenz hin zu einfacheren Wohnquartieren und Gewerbebetrieben nachvollziehen.[234]

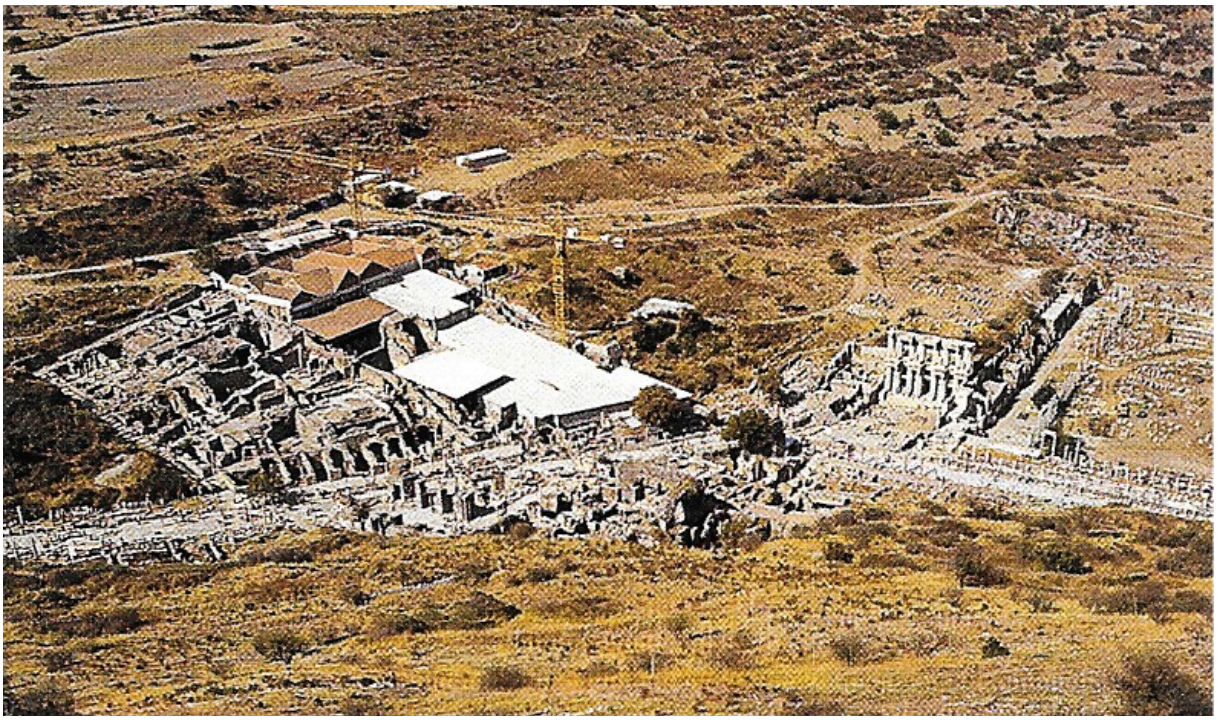


Abbildung 168 Die Hanghäuser

[233] vgl. Scherrer 1995, S. 180.
[234] vgl. Letzner 2010, S. 75.

6.2.1 Hanghaus 1

Dieses Hanghaus beinhaltet insgesamt sechs Wohneinheiten, die sich auf einer Fläche von ca. 3.000 m^2 über vier Terrassen verteilen. Weiters ist für den Komplex des Hanghauses 1 eine Nutzungsdauer vom 1. Jh. v. Chr. bis in das 7. Jh. n. Chr. festzustellen.[235]

Die einzelnen Wohneinheiten wurden nach dem im Hellenismus gebräuchlichen Grundriss – einem zentralen Peristylhof, um den sich die Räume anordnen – angelegt. Dieser Grundrisstypus und die luxuriöse Ausgestaltung der einzelnen Wohnungen verdeutlicht die Lage in einem bedeutenden Teil der Stadt. Das zu Beginn des 2. Jahrhunderts n. Chr. darüber erbaute Banketthaus nimmt etwa die Hälfte der Insulafläche ein. Der Zugang zu diesem Haus erfolgte von der Kuretenstraße über eine Taberna (Laden), genau gegenüber dem Hadrianstempel.[236]

Allgemein ließen sich im Hanghaus 1 deutliche Anzeichen von Wohnluxus feststellen, wie z.B. Bäder, Wandverkleidungen aus Marmor, Mosaiken und Brunnenanlagen. In Folge der veränderten Ansprüche (gesteigerter Wohnluxus) und in Folge des Repräsentations-bedürfnisses wurde in der 2. Hälfte des 2. Jahrhunderts n. Chr. die Bebauung angepasst. Gegen Ende des 3. Jahrhunderts, als sich die wirtschaftliche Lage im Imperium allgemein zum Schlechteren hin veränderte und Ephesos von Erdbeben betroffen war (siehe „6.1.3 Geschichtlicher Überblick“), erfuhr auch das Hanghaus 1 einen Wandel: So wurden entlang des Embolos Läden eingerichtet, die Wohnungen wurden kleiner und im Baukomplex entstanden Werkstätten. Ab dem 5. Jh. sollte die Wohnqualität dann wieder besser werden.[237]

Interessant zu beobachten ist im Hanghaus 1 auch der Siegeszug des Christentums – so wurden zwei Räume beispielsweise in eine Kapelle umgewandelt. Ein Brunnen war an der Westseite eines dieser Räume vorhanden und wurde zu einem Becken umgestaltet, das mit einer von Pfeilern mit Kreuzverzierungen gehaltenen Platte abschließt. In der Südnische des anderen Raumes waren seitlich Pfeiler, von einfachen Doppelkapitellen mit Kreuzen bekrönt, aufgestellt und auch ein weiteres ionisches Kapitell mit kreuzverziertem Abakus wurde hier gefunden. Dieser Raum wird deshalb auch Kultraum genannt, da in unmittelbarer Nähe Fragmente zweier Mensaplatten gefunden wurden, die einen zusätzlichen Hinweis auf sakrale Nutzung geben. Am Beginn des 5. Jahrhunderts lässt sich nicht nur in den Wohnungen, sondern zum Teil auch in den Geschäften, eine Wiederbelebung der Wohnkultur erkennen, welche aber nicht annähernd die Qualität des 2. und 3. Jahrhunderts erreichen konnte. Zu den

[235] vgl. Letzner 2010, S. 75.
[236] vgl. Scherrer 1995, S. 102.
[237] vgl. Letzner 2010, S. 75.

markantesten Veränderungen dieser Zeit zählen die Kreuzgrundrisse der Räume an der Ostseite der Insula.[238]

6.2.2 Hanghaus 2

Im Vergleich zum Hanghaus 1 befindet sich das Hanghaus 2 in einem viel besseren Zustand. Hier verteilen sich insgesamt sieben Wohneinheiten auf drei Terrassen mit 4000 m^2 Grundfläche. Über die Wohneinheiten lässt sich sagen, dass diese ungefähr alle gleich groß angelegt sind und – wie schon im Hanghaus 1 – dem Prinzip des hellenistischen Peristylhauses folgen. Vergleichbar ist weiters auch die Nutzungsdauer vom 1. Jh. v. Chr. bis in das 7. Jh. n. Chr. (Nutzung nach 262 allerdings nur in den Randbereichen).[239]

Die Ausgrabung dieser, im unmittelbaren Zentrum des hellenistisch-römischen Ephesos gelegenen, Insula erfolgte durch H. Vetters in den Jahren 1967-1983. Durch die 1985 abgeschlossene Überdachung und museale Präsentation der obersten Bauterrasse erfuhr das Hanghaus 2 einen ersten abschließenden Höhepunkt (im Jahr 2000 Fertigstellung der Gesamtüberdachung). Die archäologische Erforschung des Komplexes, ebenso wie konservierungstechnische Maßnahmen zur Erhaltung der Bausubstanz und ihrer Schmuckflächen sind nach wie vor im Gange.[240]

In der Spätantike ging der Charakter gehobenen Wohnens verloren, der sich u.a. in Heizungsanlagen, hauseigenen Latrinen und dekorativen Brunnen gezeigt hatte. Des Weiteren entstanden im Westen der Insula Wassermühlen und im Nordwesten eine Werkstatt mit Steinsäge, welche ebenfalls mit Wasserkraft betrieben wurde und ein bedeutendes technikgeschichtliches Denkmal darstellt (datiert wird sie in das 6. Jh.). Die gute Wasserversorgung der Hanghäuser führte zweifellos dazu, dass sich diese Betriebe hier einnisteten.[241]

Alle Wohnungen verfügten in der Regel über mehrere beheizbare Räume, wobei Heizungssysteme teilweise auch im Obergeschoß installiert waren. Latrinen und interne Laufbrunnen wurden durch das öffentliche Verteilernetz ver- und entsorgt. Darüber hinaus sicherten auch Zisternen und in den Fels geschlagene Ziehbrunnen die Wasserversorgung der einzelnen Haushalte. Die Abwasserentsorgung wiederum erfolgte über das städtische Kanalsystem unter den Stiegengassen.[242]

---

[238] vgl. Scherrer 1995, S. 104.
[239] vgl. Letzner 2010, S. 75-77.
[240] vgl. Scherrer 1995, S. 108.
[241] vgl. Letzner 2010, S. 77.
[242] vgl. Scherrer 1995, S. 112.

Weiters weist die Ausstattung der Bausubstanz im Erdgeschoß durchwegs repräsentativen Charakter auf und ist in ihrer Vielfalt, Qualität und Geschlossenheit von herausragendem kultur- und kunsthistorischen Interesse. So spiegelt sie nicht nur den Zeitgeschmack oder künstlerischen Anspruch ihrer, für uns meist anonymen, Auftraggeber wider, sondern auch das Bedürfnis durch gewählte Thematik der figürlichen Darstellungen (wie z.B. Theaterszenen, Philosophen- und Musendarstellungen, Illustrationen zu berühmten Romanen, Mythologisches, Genreszenen, usw.).[243]

Abbildung 169 Hanghaus 2

Nun kurz zu den jeweiligen Wohneinheiten dieses Hanghauses:

Wohneinheit 1 liegt im oberen Bereich des Hangs und konnte von Süden her von der sogenannten Hanghausstraße aus betreten werden. Betritt man das Haus durch diesen Eingang, so gelangt man in ein Vestibulum (Eingangshalle), das zu einem Innenhof mit vier Säulen führt, von welchem aus sich große Teile des Hauses erschließen. Innerhalb des repräsentativen Gebäudeteils sind die Räume zum Teil nach ihrer Dekoration benannt, vor allem deshalb, da die Raumnummerierung der Ausgrabungen nicht die Identifikation erlaubt.

[243] vgl. Scherrer 1995, S. 112.

Hervorzuheben ist in dieser Wohneinheit besonders das sogenannte Theaterzimmer (siehe Abb. 170), in welchem sich Darstellungen aus den Komödien des Menander und aus den Tragödien des Euripides finden.[244] Die nach dem Muster gemalter Architekturdarstellungen aufgelöste Dekorzone zeigt in der oberen Partie der Nordwand als zentrales Bild eine mythologische Szene – möglicherweise handelt es sich dabei um Herakles im Kampf mit Acheloos. Die kleinteilige Verbauung des Nordumganges im Peristylhof ist am ehesten auf einen erhöhten Raumbedarf der beginnenden Spätantike zurückzuführen, welcher durch nicht wiederaufgebaute Obergeschoßebenen begründet sein könnte.[245]

Abbildung 170 Theaterzimmer

In die **Wohneinheit 2** gelangt man von Westen. Den Eingangsbereich bildet ein viersäuliges Atrium mit dorischen Säulen, welches im Lauf der Zeit Mittelpunkt des Wirtschaftstraktes wurde. Als neuer Mittelpunkt des Hauses diente in der späten Kaiserzeit ein neunsäuliger Peristylhof (siehe Abb. 171), welcher aufwändig dekoriert ist. So findet sich beispielsweise in einer Exedra (nischenartiger Raum, siehe Abb. 172) an der Südseite des Peristylhofs ein Glasmosaik (darauf werden Dionysos und Ariadne in einem paradiesischen Weingarten dargestellt). Ebenfalls aufwändig dekoriert war auch der Speiseraum (siehe Abb. 173). So kann man beispielsweise sehen, dass entlang der Wände eine Zone verläuft, die in Weiß gehalten ist, während das Zentrum des Raumes mit Mosaiken versehen ist. Auf dem Randstreifen standen die Klinen (Ruheliegen). Dass dieser Raum eine bedeutsame Rolle einnahm, verdeutlichen auch die anderen Dekorationselemente, wie beispielsweise mit Marmor verkleidete Nischen, welche mit kostbaren Glasmosaiken verziert sind.[246]

[244] vgl. Letzner 2010, S. 77.
[245] vgl. Scherrer 1995, S. 112.
[246] vgl. Letzner 2010, S. 78.

Abbildung 171 Peristylhof

Abbildung 172 Exedra mit Glasmosaik

Abbildung 173 Speiseraum

Die **Wohneinheiten 3 und 5** waren ursprünglich eine Einheit, sind aber im Zuge nutzungsorientierter Veränderungen diagonal getrennt worden. Sie liegen auf dem westlichen, höher gelegenen Abschnitt der mittleren Bauterrasse. Die einzelnen Räume sind relativ klein und liegen jeweils um dreiseitig ausgestaltete Innenhöfe. Ihre Ausstattung ist zum Teil von besonderem kulturhistorischen Interesse: So finden sich beispielsweise im Raum 12 Darstellungen der neun Musen, der Dichterin Sappho und des Apoll; und an der Ostwand des Innenhofes haben sich die Porträts bekannter Philosophen (Sokrates aus Athen und Cheilon von Sparta) erhalten (siehe Abb. 174).[247] Wie bereits angesprochen sind die Räume recht klein und die Innenausstattung ist nicht so aufwändig. Hervorgehoben werden aber besonders der Raum 12, das sogenannte Musenzimmer (siehe Abb. 175), und der Hof 24 (siehe Abb. 176).[248]

Abbildung 174 Philosophen als Wanddekoration

Abbildung 175 Musenzimmer – Raum 12

[247] vgl. Scherrer 1995, S. 113.
[248] vgl. Letzner 2010, S. 78.

Abbildung 176 Hof 24

Über die **Wohneinheit 4** lässt sich sagen, dass das Zentrum des Hauses ein offener Hof war (siehe Abb. 177) und zahlreiche Umbauten den Bereich betroffen haben. Die nachhaltigste Veränderung war wohl, dass der Apsidensaal (halbkreisförmiges Gewölbe) der Wohneinheit 6 in die Strukturen der Wohneinheit 4 eingriff. Dass beide Wohneinheiten einen gemeinsamen Besitzer hatten, legt der Nachweis einer Treppe zwischen diesen beiden Wohneinheiten nahe.[249] Im Rahmen einer gewissen Zeitspanne (2./3. Jh. n. Chr.) fungierte diese Einheit möglicherweise als Wirtschaftstrakt für die Wohneinheit 6. Der ältesten erhaltenen Ausstattungsperiode sind als Wandmalereien die berühmte Darstellung des Sokrates (siehe Abb. 178) und die Darstellung der Muse Urania zuzuordnen.[250]

Abbildung 177 Pfeilerhof

[249] vgl. Letzner 2010, S. 78-79.
[250] vgl. Scherrer 1995, S. 113.

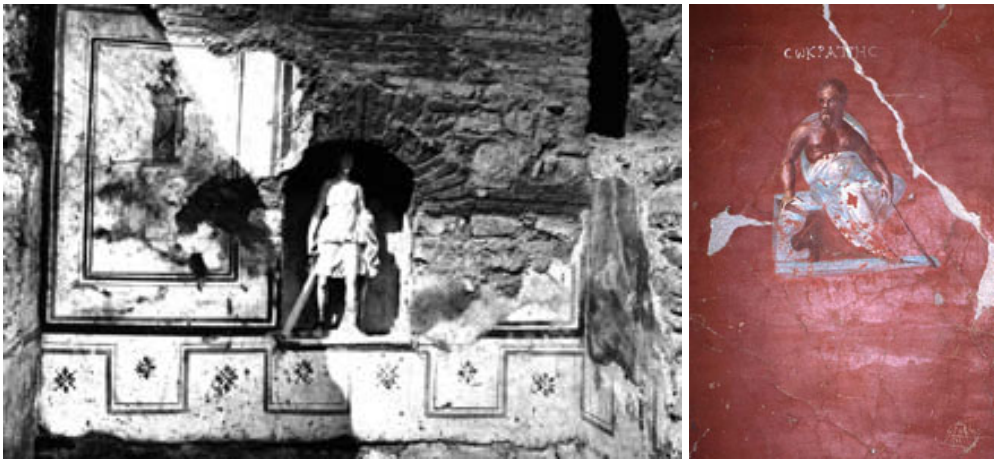


Abbildung 178 Sokrateszimmer

Die **Wohneinheit 6** ist wohl der bedeutendste Komplex des Hanghauses 2. Dafür spricht nicht nur die Größe der Wohneinheit (950 m^2), sondern vor allem der große Apsidensaal 8, durch welchen die Wohneinheit besonders hervorstach: Dabei handelt es sich um einen großen Empfangsraum (auch basilica privata oder Marmorsaal genannt, siehe Abb. 179), welcher eine marmorne Wandverkleidung und eine Stuckdecke mit dionysischen Themen besaß.[251]

Abbildung 179 Marmorsaal

Der zweigeschoßig zu rekonstruierende Peristylhof bildet den Anfang einer beeindruckenden Raumkonstellation: Von ihm aus konnten nämlich einerseits die auf ihn orientierten Räume an der West- und Nordseite betreten werden, andererseits aber auch der beeindruckende, mit Marmorvertäfelung versehene Saal im Süden und die, bereits angesprochene, basilica privata über ein kleines, mit einem Kreuzgewölbe versehenes Atrium im Südwesten.[252]

Als Eigentümer ist aus einer Inschrift im Peristylhof ein gewisser C. Flavius Furius Aptus bekannt, welcher im ausgehenden 2. Jh. n. Chr. zum Kreis der führenden Bürger von Ephesos zählte und zumindest einmal als Festspielleiter die ephesischen Olympien ausgerichtet hat.

[251] vgl. Letzner 2010, S. 79.
[252] vgl. Scherrer 1995, S. 113-114.

Seine Funktion im Rahmen des Dionysoskultes hat auch in der dekorativen Gestaltung seiner Räumlichkeiten Niederschlag gefunden.[253]

Zu guter Letzt sei noch die **Wohneinheit 7** angesprochen, welche im Westen an die Wohneinheit 6 anschließt und zeitweise mit dieser verbunden war. Die verbaute Grundfläche von 900 m^2 ist traditionell organisiert. Das bedeutet, dass sich um den zentralen Innenhof (siehe Abb. 180) Räume gruppieren, die durchwegs einer „privaten“ Wohnsituation zuzuordnen sind. Aus der auf das Peristyl ausgerichteten Exedra im Süden des Ensembles stammt ein interessanter Beleg spätkaiserzeitlicher Verehrung des iulisch-claudischen Kaiserhauses: In ihr wurden die Büsten von Livia (die Frau des Augustus) und ihres Sohnes (der spätere Kaiser Tiberius) freigelegt. Die Zerstörung des Komplexes wird in der Regierungszeit des Kaisers Gallien (262 n. Chr.) angesetzt.[254]

Dieser Wohneinheit werden Räume zugewiesen, die nach Norden hin auf einer tieferen Terrasse liegen, die bis in die Spätantike hinein noch als Wohnräume genutzt wurden. In ihm befindet sich auch der Raum 45b mit Wandbildern aus dem 3./4. Jh.[255]

Abbildung 180 Peristylhof im Obergeschoß

6.3 Inschriften

6.3.1 Inschriften und Epigraphik allgemein

Heutzutage ist man mehr denn je von Inschriften umgeben – seien es Graffiti, die man auf Wände oder Mauern mehr oder weniger „monumental“ gekritzelt, gepinselt oder gesprüht

[253] vgl. Scherrer 1995, S. 114.
[254] vgl. Scherrer 1995, S. 114.
[255] vgl. Letzner 2010, S. 79.

sieht, Hinweis- oder Verbotstafeln, also zahllose feste Aufschriften in verschiedenster Funktion, politische Slogans auf Transparenten, Tattoos oder gar Worte, die Flugzeuge auf Schriftbändern hinter sich ziehen. Moderne Techniken, ein steigender Pluralismus der Lebensbedürfnisse und die zunehmende Anfälligkeit für Einflüsse von außen haben zu dieser Überflutung geführt, welche von dauerhafter oder zumindest längerfristiger Information, die Inschriften von jeher traditionell und überwiegend erstrebten, bis hin zu flüchtiger Beeinflussung und kürzester Lebensdauer reicht.[256]

Eine verbindliche Definition des Begriffes „Inschrift" ist nicht so ohne Weiteres zu finden und die Abgrenzung gegenüber anderen Schriftäußerungen bleibt oftmals unscharf bzw. willkürlich. Als Kriterien, die insgesamt oder zumindest zum Teil auf zahlreiche Inschriften zutreffen, wären die folgenden vier: Dauerhaftigkeit, Publizität, Monumentalität und gestaltender Formwille. Einschränkend soll hierzu allerdings angemerkt werden, dass diese aber sicherlich nicht das gesamte Spektrum der als Inschrift in Frage kommenden Denkmäler von vornherein charakterisieren können.[257]

Im Folgenden werden nun drei verschiedene Definitionsversuche einander gegenübergestellt. Elisabeth Trinkls Definition lautet wie folgt:

> Das deutsche Wort „Inschrift" bedeutet, daß eine Schrift eingeschrieben ist, daß ihre Buchstaben unter die Oberfläche des Inschriftenträgers hinabreichen. Inschriften wären also streng genommen nur jene Texte, deren Buchstaben in den Stein [bzw. das Material], auf dem sie stehen, eingemeißelt [bzw. eingeritzt] worden sind.[258]

Definitionsversuche, die auf die Herstellung bzw. auf die Vielfalt der Inschriftenträger zielten, stammen von Jean Mallon und Rudolf Kloos. Erstgenannter formulierte 1952 – die antiken Inschriften im Auge habend –, dass sich die Epigraphik mit allen graphischen Denkmälern befasse, mit Ausnahme derer, die mit Tinte auf Papyrus und Pergament geschrieben sind.[259] Letztgenannter wiederum schlug für den mittelalterlichen (und neuzeitlichen) Bereich folgende Definition vor:

> Inschriften sind Beschriftungen verschiedener Materialien – in Stein, Holz, Metall, Leder, Stoff, Email, Glas, Mosaik usw., die von Kräften und Methoden hergestellt sind, die nicht dem Schreibschul- und Kanzleibetrieb angehören.[260]

Diese „Negativdefinition" habe sich letztlich in der Praxis als brauchbar erwiesen, da sie den kleinsten gemeinsamen Nenner anspricht, auch wenn sie manche Randbereiche ausklammert,

---

[256] vgl. Koch 2007, S. 22.
[257] vgl. Koch 2007, S. 23.
[258] URL: http://homepage.univie.ac.at/elisabeth.trinkl/forum/forum0897/04inschr.htm#1, aufgerufen am 21.9.2015.
[259] vgl. Koch 2007, S. 24.
[260] Koch 2007, S. 24.

die unter den Inschriften mitberücksichtigt werden sollten. Als Beispiele werden hier Namenseinträge mit Tinte auf Altarplatten angeführt, Wandkritzeleien mit Tinte oder Farbstiften (wie es sie vor allem in der Neuzeit überaus zahlreich gibt) oder auch Texte, die man auf Pergamentblättern schrieb, die auf Holz aufgezogen und in Kirchen oder Rathäusern ausgehängt wurden. Trotz weitgehend schreibschriftlicher Herstellung und der damit verbundenen graphischen Beurteilung durch die Paläographie der Buch- und Geschäftsschriften zeigen sie doch ein Streben nach Dauerhaftigkeit und/oder öffentlicher Mittelung – und dies wiederum nähert sie im weitesten Sinn den inschriftlichen Denkmälern an.[261]

Die Bearbeitung von Inschriften auf Siegeln und Münzen würde hingegen, nach der Definition von Kloos, durchaus in das Arbeitsfeld der Epigraphik fallen. Jedoch bleiben sie de facto den alten Spezialwissenschaften Sphragistik (Siegelkunde) und Numismatik (Münzkunde) überlassen – dem Autor zufolge auch mit gutem Grund, da auch ihre Aufnahme in epigraphische Editionen und Korpuswerke im Hinblick auf ihre serielle Produktion im Normalfall unterbleibt.[262]

Wie man erkennen kann, bemühten sich soeben erörterte Definitionen um eine Abgrenzung der Epigraphik oder zumindest um eine Standortbestimmung gegenüber der Paläographie und der von ihr zu untersuchenden Schriftäußerungen. Koch meint, dass die wissenschaftliche Zuordnung der Schriftzeugnisse auf Wachstäfelchen, Papyrus, Pergament und Papier zur Paläographie und der von ihr zu untersuchenden Schriftäußerungen letztlich Sache des wissenschaftlichen Konsenses sei, der aus den Begriffen von vornherein nicht hervorgeht. So bedeutet beispielsweise „Paläographie“ – nach den griechischen Bestandteilen „παλαιός“ (palaios, alt) und „γραφή“ (graphe, Schrift) – an sich nichts Anderes als „Lehre von den alten Schriften“.[263]

Der Fachausdruck „Epigraphik“ wiederum ist keine antike Bezeichnung, sondern ein moderner Fachbegriff, welcher seit 1843 als Name der Wissenschaftsdisziplin in Verwendung ist. Er leitet sich aus dem griechischen Wort „ἐπιγραφή“ (epigraphe, eigentlich eine wortgetreue Übertragung des lateinischen „inscriptio“) ab. Er bedeutet zunächst nur „Aufschrift“, d.h. Schrift auf einem Schriftträger und fand sich in diesem Sinn seit dem spätesten 17. Jahrhundert als Ausdruck für „Inschrift“. Das lateinische Wort „inscriptio“ (das deutsche Wort „Inschrift“ ist eine Lehnübersetzung) bedeutete in der Antike „Aufschrift“

[261] vgl. Koch 2007, S. 24.
[262] vgl. Koch 2007, S. 24.
[263] vgl. Koch 2007, S. 25.

bzw. „Überschrift“, welches erst ab dem 16. Jahrhundert entscheidende Verbreitung gewann.[264]

Auch wenn also das griechische Wort „ἐπιγραφή“ („Aufschrift“) und der deutsche Begriff „Inschrift“ nicht dasselbe bedeuten, so sind Epigraphik und Inschriftenkunde allerdings gleichwertige Termini. In diesem Zusammenhang liest man bei Trinkl, dass zwar jede Inschrift zwar zugleich eine Aufschrift sei, aber nicht umgekehrt. Ohne die begriffliche Logik bzw. Unlogik dieser Gleichsetzung weiter zu hinterfragen, wird von Trinkl anschließend Folgendes festgestellt[265]:

> Inschriften sind alles Schriftliche, das nicht auf Papyrus oder Pergament geschrieben ist, gleichgültig, ob die Buchstaben mit Hammer und Meißel in Stein oder Metall eingegraben oder die Buchstaben mit Bronzelettern auf den Stein appliziert oder mit Farbe auf Wände oder Tongefäße geschrieben oder in den Putz von Wänden eingeritzt oder in den noch weichen Ton von Ziegeln und Gefäßen gestempelt worden sind.[266]

6.3.2 Inschriften von Ephesos

Ephesos hat als eines der großen urbanen Zentren der alten Welt eine reiche Fülle erstklassiger Inschriften geliefert, die Auskunft über alle Bereiche des öffentlichen und privaten Lebens geben. Bis zum heutigen Tag konnten um die 6.000 Texte und Textfragmente gefunden werden, wobei sich der thematische Bogen weit spannt: so beispielsweise von öffentlichen und privaten Abmachungen und Verträgen über Volksbeschlüsse, Ehreninschriften für hellenistische Könige und römische Kaiser, hochstehende ephesische Bürger, römische Reichsbeamte, Erlässe und Briefe von Kaisern und Statthaltern, Bauinschriften, Weihinschriften und sonstigen religiösen Texten bis hin zu Grabinschriften. Zeitlich gesehen geht die Spanne vom 6. Jh. v. Chr. bis weit ins byzantinische Mittelalter und umfasst daher rund 1.500 Jahre.[267]

---

[264] vgl. Koch 2007, S. 25.
[265] vgl. URL: http://homepage.univie.ac.at/elisabeth.trinkl/forum/forum0897/04inschr.htm#1, aufgerufen am 21.9.2015.
[266] URL: http://homepage.univie.ac.at/elisabeth.trinkl/forum/forum0897/04inschr.htm#1, aufgerufen am 21.9.2015.
[267] vgl. Knibbe 1998, S. 247.

Abbildung 181 zeigt eine dieser Steininschriften, welche im Hanghaus 2 gefunden wurde.

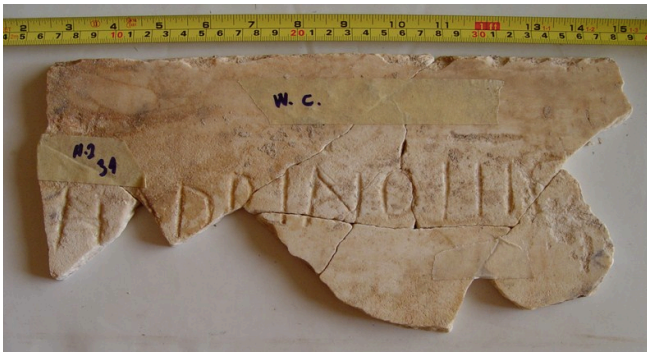


Abbildung 181 Steininschrift aus Hanghaus 2

Mit Hilfe der Inschriften ist es möglich, das politische Funktionieren der Stadt ebenso wie mehrere Tausend Ephesier aus allen Zeiten zu kennen – in der Mehrzahl jedoch aus den ersten drei Jahrhunderten der römischen Kaiserzeit, da aus dieser Zeit die mit Abstand meisten Inschriften stammen. Des Öfteren ergeben sich auch familiäre Zusammenhänge und Verwandtschaftsverhältnisse, die sich über mehrere Generationen verfolgen lassen. So wurde beispielsweise die eine oder andere Laufbahn eines römischen Reichsbeamten aus dem Ritter- und Senatorenstand erst durch eine ephesische Inschrift bekannt oder ließ sich durch eine solche vervollständigen. Inschriften auf Wandverputz (Dipinti oder Graffiti) wiederum geben Einblick in das tägliche Leben der Einwohner, in ihre Sorgen, geschäftlichen Interessen, usw. In Abbildung 182 ist eine Ehreninschrift für Damianus zu sehen, der diese wegen seiner vielen Wohltaten für die Stadt (besonders in Hinblick auf seine Verpflegung der siegreichen Armee des L. Verus mit 7.000 t Gerste) bekommen hat.

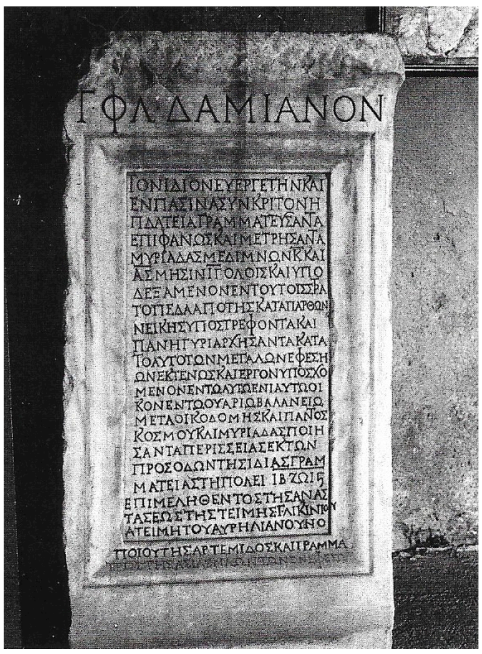


Abbildung 182 Ehreninschrift

Seit 1982 steht den Besuchern von Ephesos in den Gewölben der Terrasse des Domitiantempels auch ein kleines, im Auftrag der türkischen Antikenverwaltung vom Österreichischen Archäologischen Institut errichtetes Inschriftenmuseum zur Verfügung. Dieses bemüht sich, an Hand von rund 60 Texten (jeder Text ins Deutsche, Türkische und Englische übersetzt und erklärt) einen thematischen und zeitlichen Querschnitt durch das reiche ephesische Inschriftenmaterial zu präsentieren.[268]

Abschließend sei noch auf die Wichtigkeit von Inschriften hingewiesen: So gelangen diese nämlich direkt und ohne den Umweg über mittelalterliche Abschreiber (sozusagen als „sprechende menschliche Worte“) über die Zeiten hinweg auf dem jeweiligen Inschriftträger – Stein, Metall, Ton oder Wandverputz – zu uns. Daher werden sie als „konservierte Stimmen der Vergangenheit“ bezeichnet und weiters auch „Fleisch am Skelett der materiellen (‚archäologischen') Überreste der Antike“ genannt.[269]

[268] vgl. Knibbe 1998, S. 249.
[269] vgl. Knibbe 1998, S. 249-250.

7. Zusammenfassung

In dieser Zusammenfassung soll noch einmal dargelegt werden, was im Zuge dieser Diplomarbeit alles erarbeitet wurde.

Dies sind einerseits die vier Unterrichtsszenarien, mit welchen ich einen Beitrag für eine lebhafte Vermittlung von Informatikkonzepten anhand der Zahlenrätsel von Ephesos leisten möchte. Die jeweiligen Titel der vier Szenarios lauten wie folgt: „Grundlagen von Programmiersprachen anhand von PHP und dessen Einbindung in Webstrukturen“, „Fortgeschrittene Konzepte von Programmiersprachen anhand von PHP“, „Datenbank-grundlagen und Einbindung in PHP-Code“ und „Kryptographie: Theorie und Praxis“. Die Ausarbeitungen sind jeweils so aufgebaut, dass zuerst allgemeine Informationen gegeben werden, gefolgt von einer Planungsmatrix und Code-Beispielen. Die Theorie zu den jeweiligen Themen befindet sich in einem extrigen Kapitel („3. Technische Grundlagen“), welche nicht nur zur Erklärung für Lehrende zur Verfügung steht, sondern auch – vor allem ausgewählte Ausschnitte – für die Vermittlung im Unterricht verwendet werden können. Nicht nur die theoretischen Aspekte der technischen Grundlagen sondern auch die des geschichtlichen Hintergrundes des „Entschlüsselung der Zahlenrätsel von Ephesos“-Projekts sind nutzbar für einen interdisziplinären Unterricht (Informatik in Verbindung mit Geschichte) und befinden sich ebenfalls in einem extrigen Kapitel („6. Geschichtlicher Hintergrund“).

Als zweites, sozusagen Endprodukt, ist das fertige Programm anzusehen, welches die Entschlüsselung der Zahlenrätsel von Ephesos ermöglicht und in PHP programmiert wurde. Eingebettet in ein Web Interface ist es jederzeit über die URL http://homepage.univie.ac.at/a1008130/index.php erreichbar und wird auch in Bezug auf die Forschung vom Institut für Alte Geschichte an der Universität Wien verwendet. Es konnten damit auch bereits erste Resultate mit großem Medienecho erzielt werden.

8. Literaturverzeichnis

8.1 Monographien und Herausgeberschriften

Demmig, Thomas: MySQL lernen. Anfangen, anwenden, verstehen. München: Addison-Wesley Verlag 2003.

Herold, Helmut: Grundlagen der Informatik. 2. Auflage. München: Pearson 2012.

Karwiese, Stefan: Groß ist die Artemis von Ephesos. Die Geschichte einer der großen Städte der Antike. Wien: Phoibos-Verlag 1995.

Knibbe, Dieter: Ephesus. Geschichte einer bedeutenden antiken Stadt und Portrait einer modernen Großgrabung. Frankfurt am Main: Peter Lang 1998.

Koch, Walter: Inschriftenpaläographie des abendländischen Mittelalters und der früheren Neuzeit. Früh- und Hochmittelalter. Wien/München: Oldenbourg Verlag 2007.

Kuhlmann, Gregor und Müllmerstadt, Friedrich: SQL. Der Schlüssel zu relationalen Datenbanken. 2. Auflage. Reinbek bei Hamburg: Rowohlt Taschenbuch Verlag 2001.

Letzner, Wolfram: Eine antike Metropole in Kleinasien Ephesos. Kulturführer zur Geschichte und Archäologie. Mainz: Verlag Philipp von Zabern 2010.

Münz, Stefan: Webseiten professionell erstellen. Programmierung, Design und Administration von Webseiten. 3. Auflage. München: Addison-Wesley Verlag 2008.

MySQL AB: Das offizielle MySQL 5 Handbuch. Konfiguration, Administration, Entwicklung und Optimierung. München: Addison-Wesley Verlag 2007.

Piepmeyer, Lothar: Grundkurs Datenbanksysteme. Von den Konzepten bis zur Anwendungsentwicklung. München/Wien: Carl Hanser Verlag 2011.

Reimers, Stefan und Thies, Gunnar: PHP 5.4 & MySQL 5.5. Das umfassende Handbuch. 4. Auflage. Bonn: Galileo Press 2012.

Scherrer, Peter (Hrsg.): Ephesos – Der neue Führer. 100 Jahre österreichische Ausgrabungen 1895 – 1995. Wien: AGENS-WERK Geyer & Reisser 1995.

8.2 Links

Apsel, Mathias: HTML/Regeln/Kommentar. https://wiki.selfhtml.org/wiki/HTML/Regeln/Kommentar (6.11.2015).

Bundesministerium für Bildung: Lehrplan für Wahlpflichtfach Informatik – AHS Oberstufe. https://www.bmbf.gv.at/schulen/unterricht/lp/lp_neu_ahs_21_11876.pdf?4dzgm2 (2.12.2015).

Huisman, Brian: JavaScript Graphical / Virtual Keyboard Interface. http://www.greywyvern.com/code/javascript/keyboard (1.5.2016).

PHP Group: Allgemeine Informationen. http://php.net/manual/de/faq.general.php (2.10.2015).

Rösch, Anita: Von der Reproduktion zur Reflexion. http://diepresse.com/home/politik/innenpolitik/forumbildung/759303/Von-der-Reproduktion-zur-Reflexion (21.5.2016)

Schiemann, Thorsten: Artemis Tempel. http://www.weltwunder-online.de/antike/tempel-artemis-ephesus.htm (29.11.2015).

Trinkl, Elisabeth: Die Inschriften von Ephesos. http://homepage.univie.ac.at/elisabeth.trinkl/forum/forum0897/04inschr.htm#1 (21.9.2015).

WhatsApp Inc.: Ende-zu-Ende-Verschlüsselung. https://www.whatsapp.com/faq/de/general/28030015 (3.5.2016).

Wikipedia: PHP. https://de.wikipedia.org/wiki/PHP (6.11.2015).

9. Abbildungsverzeichnis

9.1 Abbildungen aus Kapitel 3 „Technische Grundlagen – Verwendete Technologien“:

9.2 Abbildungen aus Kapitel 4 „Entschlüsselung der Zahlenrätsel von Ephesos“:

Folgende Abbildungen wurden, wenn nicht explizit anders angeführt, eigenständig erstellt:

9.3 Abbildungen aus Kapitel 5 „Unterrichtsszenarien“:

Folgende Abbildungen wurden, wenn nicht explizit anders angeführt, eigenständig erstellt:

9.4 Abbildungen aus Kapitel 6 „Geschichtlicher Hintergrund“:

Abb. 157: Die Überreste des einstigen Weltwunders. Karwiese 1995, S. 190.

Abb. 158: Geographische Lage von Ephesos. Letzner 2010, S. 6.

Abb. 159: Plan von Ephesos mit den drei Hügeln (v.l.n.r. Bülbül Dağ, Panayır Dağ und

Abb. 160: Statue der Artemis. http://www.weltwunder-online.de/fullsize/artemis-2.jpg (29.11.2015).

Abb. 161: Rekonstruktion des ersten Artemis Tempels. http://theruinsofephesus.com/history-of-ephesus/ (21.9.2015).

Abb. 162: Rekonstruktion des Artemisions. http://www.weltwunder-online.de/fullsize/artemis-5.jpg (29.11.2015).

Abb. 163: Rekonstruktion des Platzes vor der Celsus-Bibliothek. http://www.onlineephesustravel.com/ephesusterracehousestours6c.jpg (29.11.02015).

Abb. 164: Celsus-Bibliothek – Vorderansicht. Karwiese 1995, S. 207.

Abb. 165: Das Südtor der Agora – das Tor des Mazaeus und Mithridates. Karwiese 1995, S. 198.

Abb. 166: Das große Theater. Karwiese 1995, S. 193.

Abb. 167: Blick auf den Hafen. Letzner 2010, S. 100.

Abb. 168: Die Hanghäuser. Karwiese 1995, S. 213.

Abb. 169: Hanghaus 2. Letzner 2010, S. 74.

Abb. 170: Theaterzimmer. http://www.oeaw.ac.at/antike/index.php?id=98#c685 (21.9.2015).

Abb. 171: Peristylhof. http://www.oeaw.ac.at/antike/index.php?id=99#c693 (21.9.2015).

Abb. 172: Exedra mit Glasmosaik. http://www.oeaw.ac.at/antike/index.php?id=99#c693 (21.9.2015).

Abb. 173: Speiseraum. http://www.oeaw.ac.at/antike/index.php?id=99#c693 (21.9.2015).

Abb. 174: Philosophen als Wanddekoration: Scherrer 1995, S. 111.

Abb. 175: Musenzimmer – Raum 12. http://www.oeaw.ac.at/antike/index.php?id=100#c702 (21.9.2015).

Abb. 176: Hof 24. http://www.oeaw.ac.at/antike/index.php?id=100#c702 (21.9.2015).

Abb. 177: Pfeilerhof. http://www.oeaw.ac.at/antike/index.php?id=97#c670 (21.9.2015).

Abb. 178: Sokrateszimmer. http://www.oeaw.ac.at/antike/index.php?id=97#c670 (21.9.2015).

Abb. 179: Marmorsaal. http://www.oeai.at/index.php/hanghaus-2-marmorsaal.html, (21.9.2015).

Abb. 180: Peristylhof im Obergeschoß. http://www.oeaw.ac.at/antike/index.php?id=101#c716 (21.9.2015).

Abb. 181: Steininschrift aus Hanghaus 2. http://www.oeai.at/index.php/epigrafik.html (21.9.2015).

Abb. 182: Ehreninschrift. Karwiese 1995, S. 212.

10. Tabellenverzeichnis

Alle folgenden Tabellen wurden eigenständig erarbeitet:

11. Abkürzungsverzeichnis

CSS = Cascading Style Sheets

CSV = Comma-Separated Values

DBMS = Datenbankmanagementsystem

HTML = Hypertext Markup Language (Englisch für Hypertext-Auszeichnungssprache)

ID = Identifier/Identifikator

PHP = PHP: Hypertext Preprocessor (ursprünglich Personal Home Page Tools)

SQL = Structured Query Language

URL = Uniform Resource Locator (Englisch für einheitlicher Ressourcenanzeiger)

WWW = World Wide Web (Englisch für weltweites Netz)

XOR = eXclusive OR (Englisch für ausschließendes Oder)