RHEINISCHE FACHHOCHSCHULE KÖLN

University of Applied Sciences

Fachbereich: Wirtschaft & Recht

Studiengang: Wirtschaftsinformatik (B.S.)



**Projektdokumentation**

Aufbau und Design von

Pyramidenverleih.de

|  |  |  |
| --- | --- | --- |
| Thesis vorgelegt von: | Finn Plevier | 217172009 |
|  | Severin Bustian | 213172010 |
|  | Kevin Haßfeld | 213172007 |
|  | Henning Lehmann | 213172003 |
|  | Lukas Albani | 213172008 |

Prüfer: Prof. Dr. Thomas Barth

Wintersemester 2018/2019

Inhalt

[1 Einleitung 1](#_Toc534747045)

[2 Konzept 1](#_Toc534747046)

[2.1 Idee 1](#_Toc534747047)

[2.2 Produkte 1](#_Toc534747048)

[2.3 Kunden 1](#_Toc534747049)

[2.4 Bestellung 1](#_Toc534747050)

[3 Datenbank 2](#_Toc534747051)

[3.1 ERD-Modell 2](#_Toc534747052)

[3.2 Datenbankverbindung 3](#_Toc534747053)

[3.3 Ajax 3](#_Toc534747054)

[4 Technologien 5](#_Toc534747055)

[4.1 Formulare 5](#_Toc534747056)

[4.2 Sessions 7](#_Toc534747057)

[5 Design 8](#_Toc534747058)

[Abbildungsverzeichnis 1](#_Toc534747059)

# Einleitung

Diese Projektdokumentation befasst sich mit dem Ergebnis der Zusammenarbeit von den oben genannten Studenten. Die Aufgabe war es eine Webseite mit Front- und Backend zu konzeptionieren, designen und programmieren. Die Entwicklung wird auf den folgenden Seiten erläutert. Für die Vergleichbarkeit und Bewertung sollten unteranderem folgende Funktionalitäten abgebildet werden:

* Datenbank-/ modell
* Verwendung von Formularen
* Sessions
* Suche mittels Ajax

# Konzept

## Idee

Die Idee besteht darin, einen Webshop zu entwickeln, der die Bedürfnisse eines Pyramidenverleihs deckt. So müssen zum Beispiel Besonderheiten beim Versand, der Produktvermittlung und Bezahlung berücksichtigt werden.

## Produkte

Da der Markt für den Pyramidenverleih als Nischenmarkt zu definieren ist, werden über den Webshop nicht nur Pyramiden sondern auch Sphinxen angeboten. Es soll möglich sein, seine persönliche Wunsch Pyramide zu leihen, dafür wird eine Volltextsuche benötigt. So könnte diese schnell gefunden werden. Nach der Ausführung der Suche werden die passenden Suchergebnisse angezeigt und weitere über die Suchanfrage hinaus passende Produkte. Des Weiteren sollen auf einer Detailseite weitere Informationen, wie Bilder und Preise angezeigt werden. Die Produkte und Preise sind für alle Besucher der Webseite ohne Registrierung sichtbar.

## Kunden

Um eine Bestellung auszulösen, kann sich ein Kunde mit seinen persönlichen Daten registrieren. Er kann sich auch direkt vor dem Bestellprozess registrieren. Bereits registrierte Kunden müssen sich zum Bestellabschluss einloggen. Dies geschieht mittels Benutzername und Passwort, die an seinem Benutzerkonto gespeichert sind. Die Kunden geben ihre Adresse bei der Registrierung oder Bestellung an. Die Rechnungsadresse ist die Lieferadresse.

## Bestellung

Je Bestellung kann nur ein Produkt bestellt werden. Die Bestellung erfolgt unter Angabe des Leihzeitraumes. Aus dem Preis pro Tag und der Anzahl der Tage wird der Endpreis ermittelt. Die Versandkosten sind bereits mit einkalkuliert. Es soll jedoch später auch möglich sein mit einer Bestellung mehrere Produkte für einen Zeitraum zu bestellen. Eine Bestellbestätigung und Rechnung soll an die hinterlegte E-Mailadresse versendet werden. Dies geschieht augenblicklich nach Bestellabschluss. Ein Kunde soll nach Bestellung eingeloggt bleiben, um weitere Bestellungen tätigen zu können.

# Datenbank

Um eine komplette Webshop Umgebung darzustellen, wird die Software XAMPP eingesetzt. In dieser ist neben einem Apache Webserver auch ein Mysql-Server integriert. Daher basiert das Datenbankschema auf ein relationales Datenmodell.

## ERD-Modell

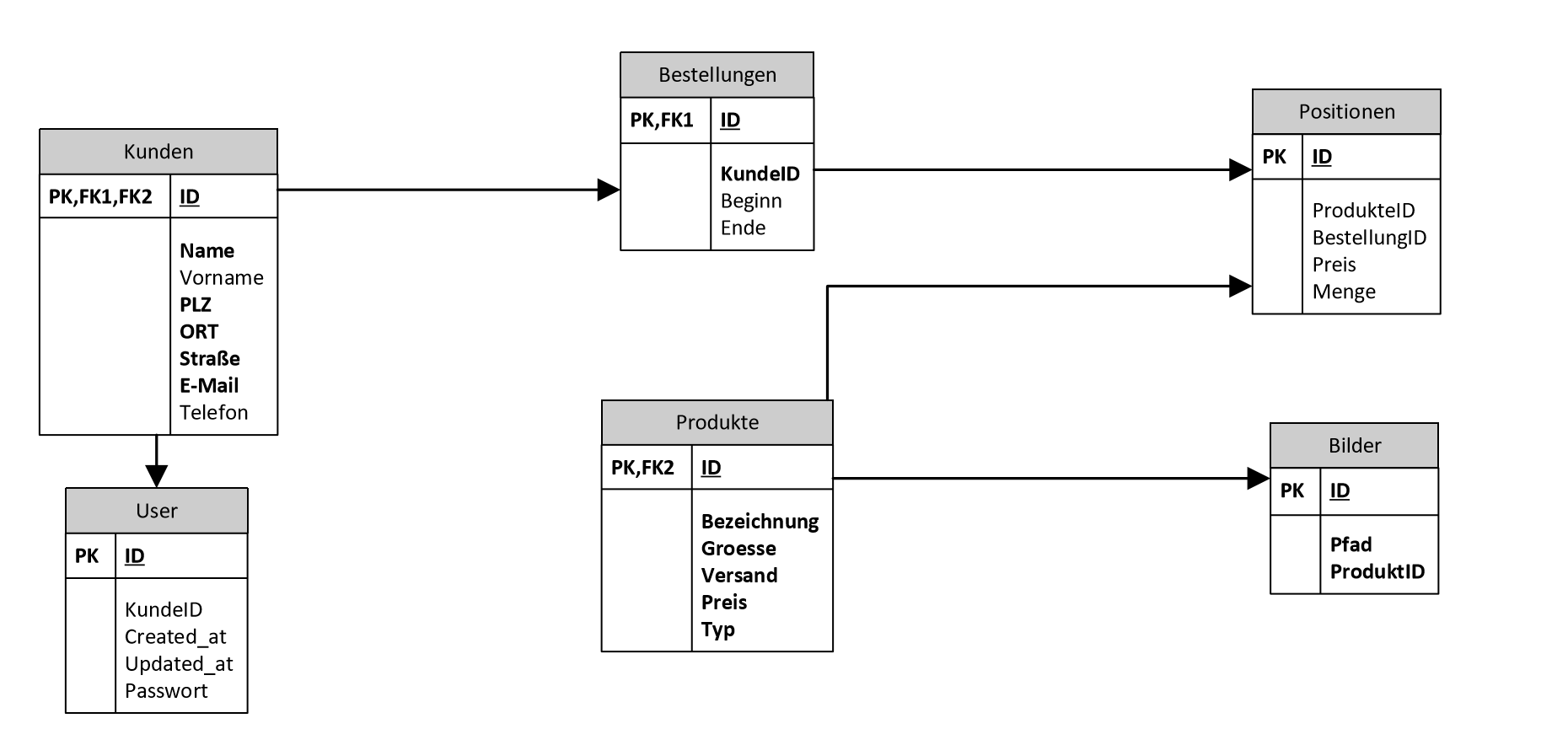


Abbildung 1: ERD-Modell

Das Herz jedes Webshops sind die Produkte, die darin angeboten werden. Daher geht von der Tabelle Produkte die meisten Beziehungen aus. In dieser Tabelle werden die grundlegenden Artikelinformationen wie Bezeichnung, Größe, Versand (in Sklaven) sowie der Preis gespeichert. Da der Webshop sowohl Pyramiden als auch Sphinxe anbietet, gibt es zusätzlich noch eine Typunterscheidung der Produkte. Um jedes Produkt eindeutig identifizieren zu können, wird außerdem für jedes Produkt eine ID vergeben, die außerdem der Primärschlüssel ist.

Da ein Produkt mehrere Bilder besitzen kann, werden diese in einer separaten Tabelle gespeichert. Die Bilddateien werden nicht in der Datenbank gehalten, sondern in einem extra Verzeichnis abgespeichert. In der Tabelle wird lediglich der Pfad zur Bilddatei sowie ein Fremdschlüssel zur ID des Produktes.

Beim Bestellprozess gibt es die Möglichkeit, mehrere Produkte zu leihen. Um dies dennoch in einer Bestellung erfassen zu können, werden die einzelnen Positionen in einer zusätzlichen Tabelle gespeichert. Die beinhaltet neben den Fremdschlüsseln zur Bestellung und zum Produkt noch die Menge sowie erneut der Preis. Dadurch können für einzelne Bestellungen Rabatte gegeben werden, ohne den Grundpreis des Artikels ändern zu müssen. Die Bestellungstabelle enthält neben dem Primärschlüssel das Start- sowie Enddatum für die Leihe. Durch die Differenz der beiden Daten kann dadurch der Gesamtpreis berechnet sowie die Verfügbarkeit überprüft werden.

In der Tabelle Kunden werden sämtliche Kundeninformationen gespeichert, die beim Bestellprozess benötigt werden. Eine Registrierung ist im Webshop jedoch freiwillig, weshalb es eine zusätzliche Tabelle für die User gibt. Diese besitzt einen Fremdschlüssel zu den Kundendaten. Außerdem wird das Passwort als Hashwert, das Registrierungsdatum sowie das Datum, an dem zuletzt Daten des Kunden geändert wurden. Kundendaten werden nur einmalig gespeichert, selbst wenn dieser jedes Mal als Gast bestellt.

## Datenbankverbindung

Die Daten werden grundsätzlich über PHP abgerufen bzw. bearbeitet. Dabei gibt es eine Datei *DBkonfiguration.php*, in der zentral alle Anmeldedaten definiert werden. Alle Abfragen sowie Datenmanipulationen werden mit dem root-User durchgeführt.

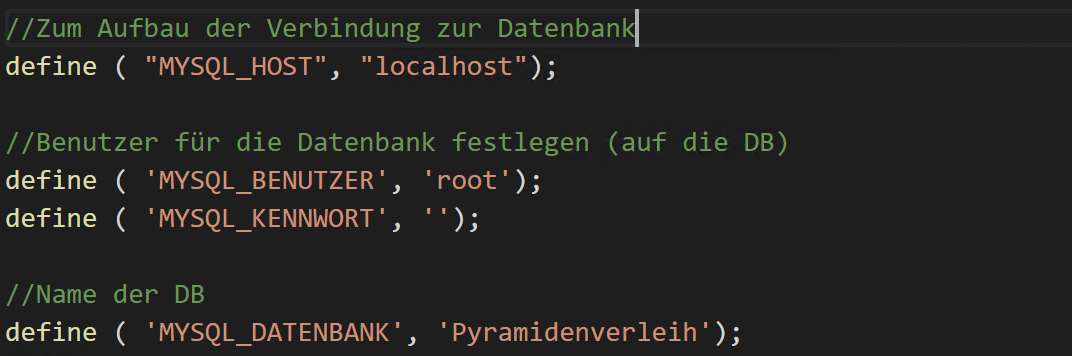


Abbildung 2: Datenbankverbindungsinformationen werden in zentraler Datei definiert

Der Verbindungsaufbau funktioniert dann auf jeder php-Seite gleich. Es wird ein PDO-Objekt erzeugt, wodurch die Datenbankverbindung hergestellt wird. Daraufhin wird ein SQL-Statement vorbereitet, was im nächsten Schritt ausgeführt wird. Anschließend können die Ergebnisse des SQL-Statements verarbeitet werden.

Als Beispiel dient hier die Produktdetailseite. Die ID des Produkts wird in der URL mitgegeben (z.B. localhost/detail.php?id=1), womit dann ein Filter in der SQL-Abfrage möglich ist, womit nur Daten selektiert werden, die zu dieser ID gehören. Die Bilder werden mithilfe der php-Funktion *getImage()* abgerufen, wobei sowohl die ProduktID als auch die Anzahl der Bilder mitgegeben werden. Dadurch kann die Funktion auf allen Seiten genutzt werden, ohne gleich sämtliche Bilder, die zu dem Produkt gehören, darstellen zu müssen.

## Ajax

Anfragen an den Webserver werden üblicherweise über Formulare getätigt. Dies hat zur Folge, dass der Webserver die Antwort jedes Mal auf einer neuen Seite darstellen muss, was zu einem trägem und nicht innovativen Benutzererlebnis führen kann. Man stelle sich eine Suchleiste vor, die auf jedes Zeichen, das der Benutzer eingibt, reagiert. Dadurch müsste der Server für jedes Zeichen eine neue Seite erstellen, was zu erheblicher Last führen kann. Um dies zu verhindern, kann Ajax eingesetzt werden.

Ajax steht für Asynchronous Javascript and XML. Es übernimmt stellvertretend für den Benutzer die Erstellung von Abfragen an den Server mithilfe von XML-Dokumenten. Dieses enthält nur Daten, die tatsächlich für die Serveranfrage benötigt werden. Da diese Anfrage asynchron ist, kann der Server diese verarbeiten, während der Benutzer weiterhin mit der Webseite interagieren kann. Ist die Antwort vom Server bereitgestellt worden, kann diese dann mit vorgefertigten Javascript-Funktionen in das HTML-Dokument integriert werden.

Als Basis dafür wird zunächst ein XmlHttpRequest-Objekt erzeugt. Mithilfe der Methode *open()* wird die Anfrage definiert. Dazu wird neben dem Skriptnamen, welches ausgeführt werden soll auch der Aufruftyp Get sowie der Suchparameter, der im PHP-Skript dann verarbeitet wird, angegeben. Letzteres wird mit *?parameter=Wert*  ans Ende des php-Skriptnamens geschrieben. Im nächsten Schritt wird dieses an den Server gesendet.



Abbildung 3: Javascript-Code für Ajax

In diesem Programmcode wird nun solange gewartet, bis der Request vom Server erfolgreich abgeschlossen wurde. Indikatoren dafür sind zum einen der *ReadyState* (4 = Request abgeschlossen und Antwort bereit) als auch der *state*, der den http-Status zurückgibt (200 = Alles ok). Sind beide Bedingungen erfüllt, wird die Serverantwort in einem div auf der Webseite dargestellt.

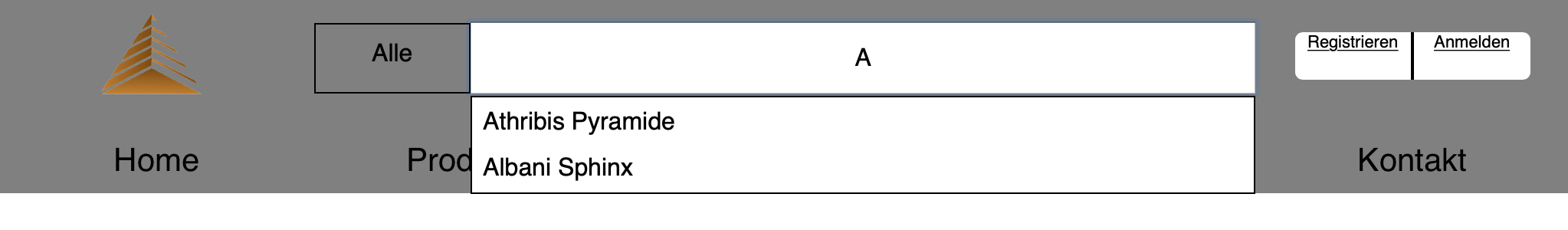


Abbildung 4: Suchvorschläge mithilfe von Ajax

Die Webseite ist so konfiguriert, dass bei jedem Event *onkeyup*, also jedes Mal, wenn im Suchfeld eine Taste losgelassen wird, die Javascript-Funktion ausgelöst wird, die für Ajax benötigt wird. Somit ändern sich die Suchvorschläge bei jedem Buchstaben, der eingetippt wird.

# Technologien

## Formulare

Um Funktionalitäten, wie zum Beispiel den Login und die Registrierung, umzusetzen werden Formulare benötigt, wo Daten eingegeben werden können. Diese können anschließend per GET- oder POST-Methode weiter versendet werden.

GET und POST sind Anfragenmethoden für das HTTP. Mithilfe von ihnen können die Browser des Clients Informationen an den Server versenden, Daten anzufordern sowie Formulare abzusenden.

Die GET-Methode wird verwendet, wenn Daten von einer spezifizierten Quelle abgefragt werden soll, während die POST-Methode verwendet wird, wenn Daten an einen Server versendet werden sollen.

In HTML wird per <form>-Befehl ein Formular angelegt, dass mehrere <input>-Felder enthalten kann.



Abbildung 5: Registrieren.php Formular

Als Beispiel dient hier die Registrieren.php. Die eingegeben Daten werden per POST-Methode an die eigene Seite gesendet. Dabei wird auch noch der GET-Parameter „register=1“ mitübergegeben, damit überprüft werden kann, ob Daten für die Registrierung gesendet wurden.



Abbildung 6: Registrieren.php

Nachdem geprüft wurde, ob der GET-Parameter übergeben wurde, muss überprüft werden, ob die Eingaben korrekt sind. Es wird daher überprüft, ob die E-Mail gültig ist und noch nicht registriert wurde. Dazu wird untersucht, ob in der Kundentabelle die E-Mail bereits existiert, da man als Gast nicht zwingend angemeldet sein muss. Wurde eine E-Mail gefunden, dann wird der Eintrag aktualisiert.

Sollten Fehler auftauchen, wird die Variable $error auf true gesetzt. Sind keine Fehler vorhanden, werden die ID und das eingegebene Passwort in die Tabelle User eingetragen.

Anschließend darf der Benutzer auf den Link klicken, um auf die Login Seite zu gelangen. Dort hat er die Möglichkeit sich mit seinem Account anzumelden.

Dazu wird überprüft, ob bereits eine Sessionvariable existiert, die eine „userid“ enthält. Wenn keine aktiv ist, dann wird das Anmeldeformular angezeigt, mit den man sich anmelden kann. Die eingegebenen Daten werden per POST-Methode an die eigene Seite versendet. Zusätzlich wird der GET-Parameter „login=1“ mitgegeben, um zu überprüfen, ob die Daten abgeschickt worden sind. Anschließend wird überprüft ob die E-Mail existiert und ob das Passwort übereinstimmt.

Sollte der Login erfolgreich sein, wird eine Sessionvariable initialisiert, namens „userid“. Anschließend wird ein Link angezeigt, der auf die Homepage verweist. Sollte man bereits eingeloggt sein, dann erscheint statt dem Anmeldeformular die Nachricht, dass man bereits angemeldet sei, und man sich per Klick auf dem Button „Ausloggen“ sich ausloggen kann.

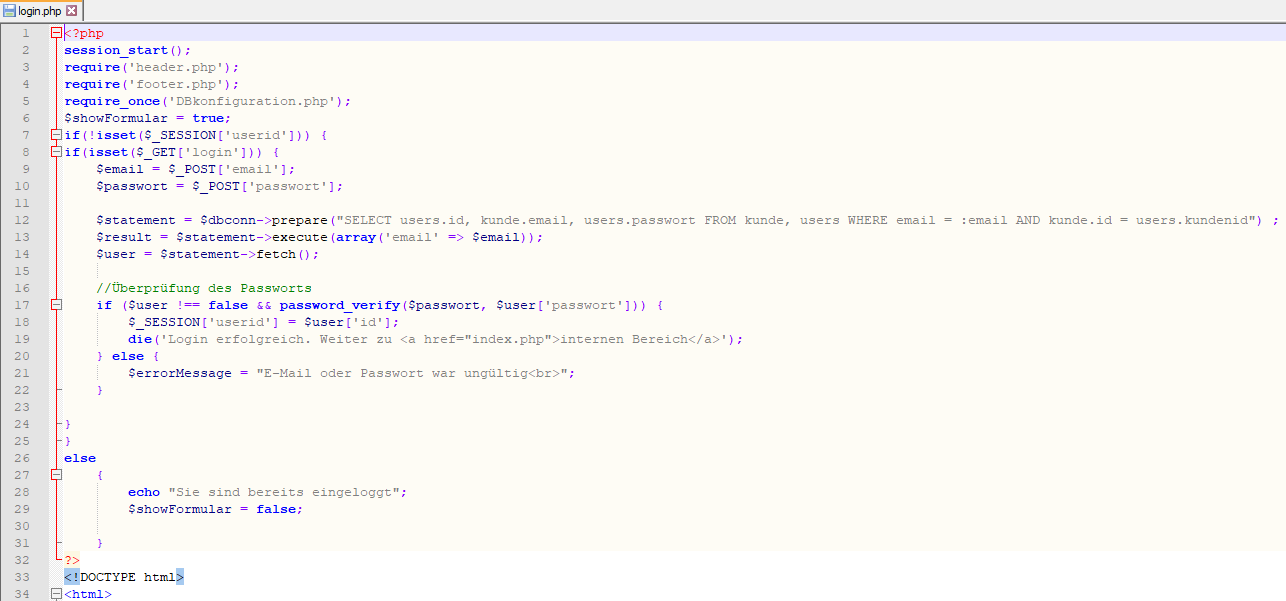


Abbildung 7: Login.php

Sollte ein Kunde sich angemeldet haben und einen Gegenstand einkaufen wollen, dann werden die Daten bei der Bestellung (bestellung.php) erscheinen. Diese sind korrigierbar, falls die Adresse abweichen sollte. Anschließend kann man auswählen, wie lange die Pyramide bzw. Sphinx geliehen werden soll.

Schlussendlich werden per Klick auf den Button „Bestellen“die eingegebenen Daten per POST-Methode an die Abschluss.php versendet.

## Sessions

HTTP ist ein Protokoll, welches keine Informationen zwischen den verschiedenen Aufrufen des Benutzers speichert. Dies ist unpraktisch, wenn mehrere Informationen gespeichert werden sollen, damit sich zum Beispiel der Benutzer anmelden kann.

Dafür gibt es Sessions, die bestimmte Informationen über mehrere Aufrufe hinweg speichern. Im Pyramidenverleih tauchen diese daher bei der Registrierung, Login und beim Bestellen auf.

So wird beim Login überprüft, ob man bereits angemeldet wird. Dies geschieht mithilfe der „userid“, die beim erfolgreichen Login erstellt wird. Sollte man bereits angemeldet sein, wird das Anmeldeformular nicht angezeigt und die Option zum „Ausloggen“ wird angezeigt.

Die „userid“ wird über mehrere Seiten hinweg gespeichert, weshalb man, wenn man eine Sphinx bzw. Pyramide leihen will, beim Bestellvorgang seine eingegebenen Daten (welche man bei der Registrierung eingegeben hat) wiederfindet.

# Design

Abbildungsverzeichnis

[Abbildung 1: ERD-Modell 2](#_Toc534747035)

[Abbildung 2: Datenbankverbindungsinformationen werden in zentraler Datei definiert 3](#_Toc534747036)

[Abbildung 3: Javascript-Code für Ajax 4](#_Toc534747037)

[Abbildung 4: Suchvorschläge mithilfe von Ajax 4](#_Toc534747038)

[Abbildung 5: Registrieren.php Formular 5](#_Toc534747039)

[Abbildung 6: Registrieren.php 6](#_Toc534747040)

[Abbildung 7: Login.php 7](#_Toc534747041)