

Diplomarbeit vom 6. September bis 29. Oktober 2001

Studenten: José Fontanil und Reto Glanzmann

Dozent: Patrick Feisthammel

PhPepperShop - Architektur



Architektur

Dieses Dokument ist ein Auszug aus der kompletten Diplomarbeitdokumentation. Hier werden lediglich architektonische Aspekte des PhPepperShops erklärt. Weitere, zum Teil auch die Architektur betreffende Informationen, findet man in der Dokumentation 'PhPepperShop – Bericht', Kapitel Erfahrungen.

Inhaltsverzeichnis

1 Security

1.1 PHP.....	4
1.2 htaccess.....	4
1.3 Datenbank (im Speziellen MySQL).....	4
1.4 SSL.....	5
1.5 GNU-PG.....	5
1.6 Kunden-ID.....	6

2 PHP-Module und -Funktionen

2.1 Übersicht.....	7
2.2 PHP-Module und ihre Funktionen.....	9
Klassendefinitionen.....	9
2.3 Fehlerbehandlung in den Modulen.....	39
Allgemeine Sicherheitsvorkehrungen.....	39
Fehlercode.....	39
Beispiele.....	39

3 Session Management

3.1 Maximale Session Dauer.....	41
3.2 Session-Management-Header.....	41

4 Datenbank

4.1 Idee und Konzept.....	43
4.2 Personalisierung durch das Installationstool config.pl.....	43
4.3 Relationen in der Datenbank (Übersicht).....	44
4.4 Entity Relationship Diagramm der Shop Datenbank.....	44
ER Diagramm Teil 1.....	45
ER Diagramm Teil 2.....	46
ER Diagramm Teil 3.....	47
4.5 Tabellen und ihre Attribute im Detail.....	48
4.6 SQL-Query Handling.....	50
4.7 DB-Wrapper (database.php).....	51

5 Shop Layout Management

5.1 Shop Layout Beschreibung	52
5.2 Layout Administrationsmodus.....	52
5.3 Layout Management.....	53
Layout Management.....	53
Hintergrundbilder und Shoplogo hochladen.....	54
Buttons hochladen.....	54
5.4 Buttons im Administrationsbereich.....	56
5.5 Ändern der Welcome Page.....	57

6 Artikelmanagement

6.1 Artikelsuche.....	60
Mehrere Suchbegriffe in einer Abfrage erlaubt.....	60
Bilder können optional angezeigt werden.....	60
Mehrfachkategorien Anzeige.....	60
Inkrementangabe.....	60
Screenshot.....	61

7 Kategorienmanagement

7.1 Ursprüngliches Konzept (unverändert).....	62
Kategorienmanagement Bildschirmsicht:.....	62
neue Kategorie erstellen [1]:.....	63
Kategorie verschieben [2]:.....	63
Kategorie umbenennen [3]:.....	63
Kategorie löschen [4]:.....	63
neue Unterkategorie erstellen [5]:.....	63
Unterkategorie innerhalb Kategorie verschieben [6]:.....	64
Unterkategorie in eine andere Kategorie verschieben [7]:.....	64
Unterkategorie umbenennen [8]:.....	64
Unterkategorie löschen [9]:.....	64
7.2 Hauptmenü.....	65

8 Zahlungsmöglichkeiten

8.1 Vorkasse, Rechnung, Nachnahme und kostenlos.....	66
8.2 Kreditkarten.....	66
Shop-intern abgewickelte Kreditkartenzahlung.....	66
Einschränkungen.....	66
Shop-extern abgewickelte Kreditkartenzahlung.....	66
Idee und Konzept.....	67
Usability der Schnittstelle.....	67
Das Testmodul (pay_ext_test.php).....	68

9 Bestellungsmanagement

10 Kundenattribute Management

11 CRM – Customer Relationship Management

11.1 Welche Funktionen bietet das CRM.....	71
--	----

12. Anhang

12.1 Literaturverzeichnis.....	73
12.1.1 PHP.....	73
12.1.2 Apache Webserver.....	73
12.1.3 CRM – Customer Relationship Management.....	74
12.1.4 Datenbanken und im speziellen MySQL.....	74
12.1.5 HTML, JavaScript und Cascading Style Sheets.....	74
12.1.6 Perl.....	75
12.2 Software Versionen.....	76
12.2.1 Auf dem Server eingesetzte Software.....	76
12.2.2 Auf den Arbeitsstationen eingesetzte Software.....	76
12.3 Zusatzdokumente.....	76

Architektur

1 Security

Wir haben uns viele Gedanken betreffend der Security des Shops gemacht und Sicherheitsmassnahmen auf mehreren Ebenen eingebunden. Im Folgenden gehen wir genauer auf die einzelnen Teile der Shop-Security ein.

1.1 PHP

Da ein potentieller Hacker/Cracker den Sourcecode der HTML-Seiten sowie Header-Informationen begutachten und verändern kann, besteht die Gefahr, dass er sich so ein Bild über den Shop-Aufbau machen kann und auf diese Weise zu wichtigen Hintergrundinformationen kommen könnte.

Um dem vorzubeugen, haben wir uns in der ersten Projektarbeit schon beim Design dafür entschieden, diejenigen PHP-Scripte, welche Administrations-Aufgaben wahrnehmen, von den USER-Level-Scripten so gut es geht zu entkoppeln.

Dies fängt schon bei den SQL-Kommandos an: Alle SQL-Befehle, die von USER-Funktionen benutzt werden, kommen aus einer Datei. Alle Administrator-Scripts befinden sich in einer anderen Datei, die von keinem USER-Modul included wird. Auf diese Weise ist sichergestellt, dass von einer USER-Funktion niemals ein Administrations-Script aufgerufen wird.

PHP bietet einen sogenannten SafeMode an. Durch ihn versucht PHP einige Sicherheitslücken zu schliessen. Dieser Modus beinhaltet ein Sicherheitskonzept, welches die Zugriffsberechtigung auf Dateien regelt. Ein Zugriff auf eine Datei wird nur noch gewährt, wenn die Datei, das Verzeichnis und das Script dem gleichen Benutzer gehören. Dieses Kriterium gilt weiter für alle Funktionen, die ein Risiko für die Systemsicherheit darstellen könnten. Weiter kann man ausführbare Dateien nur ausführen, wenn sie sich im `save_mode_exec`-Verzeichnis befinden. Dieses kann in der Datei `php.ini` definiert werden. Wir haben den SafeMode selbstverständlich eingeschaltet.

Damit ein Benutzer nicht eine Bestellung mit manipulierten Preisen aufgeben kann, werden diese immer aus der Datenbank geholt und an das jeweils anfordernde Script übergeben.

1.2 htaccess

Der Administrations-Teil wird weiter per htaccess geschützt. Alle Administrations-Scripte sind im Unterverzeichnis `Admin` abgelegt. Wer einen anderen Server (z.B. IIS) einsetzt, muss dafür sorgen, dass das nur vom Administrator auf das Verzeichnis `Admin` zugegriffen werden kann.

1.3 Datenbank (im Speziellen MySQL)

Um dieses Konzept auch auf der Datenbank-Ebene fortzusetzen, arbeitet der Shop grundsätzlich mit zwei Datenbank Benutzern. Einerseits mit dem `shopuser` und um die Administrator-Scripte auszuführen als `shopadmin`. Dem `shopuser` wurden nur die nötigsten Rechte gewährt. Er kann den Grossteil der Datenbank nur lesen. Bestellungen darf er updaten und löschen.

MySQL selbst wird auch nicht als `root` gestartet, sondern mittels der Anweisung:

```
../mysql/bin/safe_mysqld --user mysql&
```

Auf diese Weise wird MySQL von einem User gestartet, welcher analog zum Apache-User nur über sehr eingeschränkte Rechte im UNIX Betriebssystem verfügt.

Zum Schluss stellten wir noch sicher, dass die beiden User shopuser und shopadmin keine GRANT- und ALTER-Rechte haben.

In dem von vielen Personen gewünschten Ein-User-Betrieb des Shops, greifen die MySQL-Security Überlegungen nicht. Der Shop erledigt alle Operationen mit dem shopdamin-User.

Man kann den PhPepperShop auch mit einem (oder zwei) schon bestehenden Datenbank-Usern betreiben. Hier wissen wir natürlich nicht, welche Rechte diesen Usern gegeben wurden. Wir haben diese Flexibilität eingebaut, weil es viele Anwender gibt, welche an ihrem vom Hosting-Provider erhaltenen MySQL-Account nichts ändern dürfen. Man sollte sich aber im Klaren darüber sein, dass, wenn diesen Benutzern GRANT-/ALTER-Rechte oder sogar Rechte an weiteren Datenbanken gewährt werden, dies eine potentielle Sicherheitslücke darstellt.

1.4 SSL

Da wir im PhPepperShop Kreditkartenhandling eingeführt haben, müssen diese sensiblen Daten auch verschlüsselt übertragen werden. Dafür bietet sich SSL an. Wir haben SSL auf unserem Webserver installiert und zu Testzwecken mit einem selbst erzeugten Serverzertifikat betrieben. Damit ein Shopbenutzer dem Shop aber vertrauen kann, sollte man hier ein offizielles Serverzertifikat, z.B. von VeriSign, benutzen.

Der Shopadministrator kann in den allgemeinen Shopeinstellungen SSL optional einschalten. Es ist ihm überlassen, ob er SSL auf dem Webserver installieren will oder nicht. Wir empfehlen die Verwendung von SSL, sobald jemand Kreditkarten als Zahlungsmöglichkeit anbieten will. Man muss sich aber auch bewusst sein, dass SSL-Verbindungen 'langsam' sind.

Es wird beim PhPepperShop nicht der komplette Frameset über HTTPS übertragen, sondern nur der Content-Frame (welcher die Formulare enthält). Aus diesem Grund zeigen Netscape 4.7x und der Internet Explorer auch kein 'sicheres Schlösschen' an. Dies ist verwirrend. Wenn sich jemand überzeugen will, ob seine Eingaben wirklich SSL geschützt sind, so kann er dies folgendermassen tun:

- Mozilla / NS 6.x: Zeigt auch hier das 'Sicherheits-Schlösschen' an.
- Internet Explorer: Rechts-Klick -> Eigenschaften -> dort muss https als Protokoll stehen.
- Netscape 4.7x: Rechts-Klick -> Informationen anzeigen -> dort muss https als Protokoll stehen.

1.5 GNU-PG

Eurocard und VISA bieten einen lukrativen MailOrder Vertrag an, sobald man nachweisen kann, dass die Kreditkartendaten der Kunden nie unverschlüsselt übertragen werden. Damit dies gewährleistet ist, bietet sich eine Kombination aus SSL und GNU-PG an. Die Kreditkartendaten werden per SSL-verschlüsselt vom Kunden an den Shop übermittelt. Das Bestellungs-E-Mail für den Shopadministrator wird dann vor dem Versand mit GNU-PG oder PGP verschlüsselt. Da die Kreditkartendaten nicht in die Shop-Datenbank gespeichert werden, können sie nie unverschlüsselt abgerufen werden. Nur der Shopbetreiber kann sie mit seinem privaten Schlüssel wieder entschlüsseln.

Aufgrund, der in den Erfahrungen erwähnten Problemen, ist die E-Mail Verschlüsselung mittels GNU-PG leider noch nicht implementiert worden.

1.6 Kunden-ID

Damit die Kunden identifiziert werden können, erhält jeder eine Kunden_ID. Diese wird über gewisse Formulare hinweg als hidden-Field übertragen. Damit hier nicht ein Sicherheitsloch entsteht, haben wir die Kunden-ID nicht einfach als fortlaufende Integer Zahl implementiert. Sie besteht aus einer Ganzzahl, welche folgendermassen zusammengesetzt ist:

1. Mit dem PHP-Zufallszahlengenerator `rand()` erzeugen wir drei unabhängige Zufallszahlen aus dem Bereich von 1 bis `RAND_MAX` (bei uns 2'147'483'647)
2. Diese drei Zahlen werden als String aneinander gereiht und auf ihre Einmaligkeit unter den schon bestehenden Kunden_IDs hin überprüft. Es entsteht ein Zahlenraum von 111 bis 214'748'364'721'474'836'472'147'483'647. Die Kunden_ID in der Tabelle `kunde` hat einen eigenen Index und wird als `varchar(255)` gespeichert.
3. Ist die Zahl einmalig, so erhält der Kunde diese Kunden-ID.

Mit dieser Kunden-ID wird es böswilligen Personen praktisch verunmöglicht, durch Raten von Kunden-IDs an Kunden-Informationen zu gelangen, respektive unter deren Namen zu bestellen.

2 PHP-Module und -Funktionen

Im folgenden wird nun die Implementation des Shops aus struktureller Sicht genauer erläutert. Die Übersicht soll helfen, den Aufbau des Shops besser verstehen zu können. Die Modul- und Funktionsbeschreibungen geben einen Überblick über die Abbildung der Shop-Funktionalitäten auf die PHP-Skripte. Damit man den Unterschied von der Projektarbeit zur Diplomarbeit sieht, wurden geänderte und neu geschriebene Funktionen und Module mit einem **DA** gekennzeichnet.

2.1 Übersicht

Der Shop wurde funktionell in mehrere Schichten unterteilt. Die PHP-Module (Dateien) sind jeweils einer der Schichten zugeordnet.

<i>Schichtenmodell des PhPepperShops</i>
HTML-Darstellung (Frameset, Kunden-Navigation im top-Frame)
Darstellen Module (hier werden die komplexen Abläufe in Funktionen mit HTML / Plain-Text Output behandelt)
Aufruf Module (rufen Funktionen auf und generieren HTML)
Payment-Interface (Schnittstelle für externes Zahlungssystem)
PHP-Funktionen
Datenbankanbindung (Getrennte User- und Administrator Datenbankanbindung)
Datenbank Wrapper (Spricht verschiedene Datenbanken an, bietet nach oben eine einheitliche Schnittstelle für Datenbank-Funktionen)
SQL-Statements (User- und Administrator SQL-Statements zentral in je einer Datei)
Klassendefinitionen (Definitionen der komplexeren Datenstrukturen)

Es folgt nun eine Einteilung der entsprechenden Module in ihre Schichten. Weiter ist jedem einzelnen Modul eine Sicherheitsstufe zugeordnet. User Funktionen arbeiten mit User SQLs und User Datenbank Objekten. Die Includes sind so gehalten, dass kein User-Modul auf irgendwelche als Admin deklarierte Module zugreifen kann. Admin Module dürfen hingegen auch User-Module benutzen. Die Pfadangaben beziehen sich, wenn nicht anders angegeben, auf folgenden Pfad:

```
<shopdir>/shop/
```

<i>Schicht</i>	<i>Beschreibung</i>	<i>Modul-Name</i>	<i>Security</i>
<i>HTML-Darstellung</i>	Frameset	../index.php	User
	Oberer Frame	Frameset/top.php	User
<i>Darstellen Module</i>	Bestellung	USER_BESTELLUNG_DARSTELLUNG.php	User
	User Hilfe darstellen	USER_ADMIN_HILFE.php	User
	Bild aus DB anzeigen	bild_view.php	User
	Artikelmanagement	Admin/SHOP_ADMINISTRATION_ARTIKEL.php	Admin
	Admin Hilfe darstellen	Admin/ADMIN_HILFE.php	Admin
	Erstellt ein DB-Backup	Admin/ADMIN_backup.php	Admin
	Zurücklesen von Backup	Admin/ADMIN_restore.php	Admin
	Externe Zahlungsstelle	payment_interface.php	Admin
<i>Aufruf Module</i>	Artikelhandling	USER_ARTIKEL_HANDLING_AUFRUF.php	User
	Bestellung	USER_BESTELLUNG_AUFRUF.php	User
	Bestellung	USER_BESTELLUNG_1.php	User
	User PopUp anzeigen	pop_up.php	User
	Administration	Admin/SHOP_ADMINISTRATION_AUFRUF.php	Admin
	Backup/Restore Menü	Admin/SHOP_BACKUP.php	Admin
	DB-Backup des Shops	Admin/SHOP_BACKUP_f1.php	Admin
	Bestellungsmanagement	Admin/SHOP_BESTELLUNG.php	Admin
	Kundenattribute	Admin/SHOP_KUNDE.php	Admin
	Layout Management	Admin/SHOP_LAYOUT.php	Admin
	Shop Einstellungen	Admin/SHOP_VERSANDKOSTEN.php	Admin
	Versandkosten	Admin/SHOP_SETTINGS.php	Admin
	Kategorienmanagement	Admin/Shop_Einstellungen_Menu_Kategorien.php	Admin
	Administrationsmenü	Admin/Shop_Einstellungen_Menu_1.php	Admin
	Artikelmanagement	Admin/bild_up.php	Admin
	Admin PopUp anzeigen	Admin/pop_up_admin	Admin
	<i>Payment-Interface</i>	externes Payment	payment_interface.php
externes Payment/Test		pay_ext_test.php	User
<i>PHP-Funktionen</i>	Artikelhandling	USER_ARTIKEL_HANDLING.php	User
	Bestellung	USER_BESTELLUNG.php	User
	Administration	Admin/SHOP_ADMINISTRATION.php	Admin
<i>SQL-Statements</i>	User SQL-Statements	USER_SQL_BEFEHLE.php	User
	Admin SQL-Statements	Admin/ADMIN_SQL_BEFEHLE.php	Admin
<i>Datenbank-anbindung</i>	User DB Anbindung	initialize.php	User
	Admin DB Anbindung	Admin/ADMIN_initialize.php	Admin
<i>Datenbank Wrapper</i>	DB Kommunikation	database.php	User

<i>Schicht</i>	<i>Beschreibung</i>	<i>Modul-Name</i>	<i>Security</i>
<i>Klassendefinition</i>	Artikel-Klasse	artikel_def.php	User
	Attribut-Klasse	attribut_def.php	User
	Bestellungs-Klasse	bestellung_def.php	User
	Kategorien-Klasse	kategorie_def.php	User
	Kreditkarten-Klasse	kreditkarte_def.php	User
	Kunden-Klasse	kunde_def.php	User
	Versandkosten-Klasse	versandkosten_def.php	User

Alle weiteren Dateien wurden nicht ins PHP-Schichten-Modell integriert, haben aber eine Security Stufe.

<i>Beschreibung</i>	<i>Modul-Name</i>	<i>Security</i>
User Redirect	../index.html	User
Administrationtool Redirect	Admin/index.html	Admin
Datenbank und DB-User erzeugen	database/{shopname}_create.sql	Admin
Datenbank mit Default-Shop füllen	database/{shopname}_insert.sql	Admin
Datenbank und DB-User löschen	database/{shopname}_del.sql	Admin
Installation (Diese Dateien existieren nur vor der Installation)	../config.pl	Admin
	../config_part4.sh	Admin
Deinstallation	../remove.pl	Admin
CSS-Template	Admin/csstemplate.txt	Admin
Frameset Template	Admin/indextemplate.txt	Admin
User CSS Datei	shopstyles.css	User
User CSS Datei (doppelt wegen NS. 4.7x)	Frameset/shopstyles.css	User
Fixes Admin CSS-Layout	Admin/shopstyles.css	Admin
Willkommenspage	Frameset/content.html	User

2.2 PHP-Module und ihre Funktionen

Es folgen nun die nach Schichten sortierten PHP-Module. Handelt es sich um Aufruf-Module, so wird deren Funktion anhand der übergebenen Variable 'darstellen' definiert. Bei den funktional gegliederten Modulen werden die Funktionen erklärt.

Klassendefinitionen

Wir haben im PhPepperShop immer auf Klassen zurückgegriffen, wenn komplexe Datentypen bearbeitet oder übertragen (von Funktion zu Funktion) werden mussten. Der PhPepperShop ist ganz klar nicht Objekt-orientiert entwickelt. Wir benötigten die Klassen lediglich als Daten-Container.

Modul: Klassendefinitionen	
Datei	Klassenname
artikel_def.php	Artikel
Beschreibung DA	Definiert eine Klasse Artikel, um mit den vielen Attributen eines Artikels umgehen zu können. Ein Artikel-Objekt kann auch alle Optionen und Variationen (beliebige Menge) eines Artikels fassen. Auf die Variablen der Klasse wird direkt, ohne set- / get-Funktionen zugegriffen. Für die Optionen- und Variationen-Arrays gibt es entsprechende Funktionen.
artikel_def.php	Option und Variation
Beschreibung DA	Im PhPepperShop werden Artikel Optionen und -Variationen in Arrays transportiert. Zukünftige Entwicklungen für den PhPepperShop sollen aber diese beiden Klassen verwenden. Sie bilden eine Zeile der jeweiligen Tabelle (artikel_optionen, artikel_variationen) ab. Man hat auf diese Weise viel elegantere Zugriffsmöglichkeiten auf die Daten.
artikel_def.php	Artikelmitkategorien
Beschreibung DA	Diese etwas seltsam benannte Klasse wurde für den Transport von Artikeln und ihren Kategorien erstellt. Seitdem ein Artikel in mehreren Kategorien gleichzeitig sein kann, musste man einen Weg haben, beide Informationen auf einmal transportieren zu können. Diese Klasse fasst deshalb ein Artikel-Objekt und einen Array von Kategorien.
attribut_def.php	Attribut
Beschreibung DA	Definiert die Klasse Attribut. Es existiert eine Tabelle 'attribut', in welcher diese Attribute gespeichert werden. Die Attribute werden dazu verwendet, um dynamisch Kundendatenfelder konfigurieren zu können. Man kann z.B. das Eingabefeld Strasse erzeugen und diesem (Kunden-)Attribut dann mitteilen, ob es angezeigt werden soll, ob seine Eingabe überprüft werden soll, an welcher Position es angezeigt werden soll und ob es persistent zu jedem Kunden mit in der Datenbank gespeichert werden soll (CRM). In dieser Klasse wird nicht nur ein Attribut gespeichert, sondern immer gleich alle existierenden Attribute (Arrays).
bestellung_def.php	Bestellung
Beschreibung DA	Definiert eine Klasse Bestellung, die der Forderung gerecht wird, eine ganze Bestellung inklusive all ihren Attributen und Artikel-Zuordnungen aufnehmen zu können. Die Artikel werden dabei in einem Array (für den es Zugriffsfunktionen gibt) verwaltet. Dieser Array enthält Artikel_info-Objekte (siehe weiter unten).
bestellung_def.php	Artikel_info
Beschreibung	Diese Klasse wird nur im Zusammenhang mit Bestellungen verwendet. Sie enthält Informationen bezüglich eines Artikels in einer Bestellung (also auch Anzahl u.s.w.).
kategorie_def.php	Kategorie
Beschreibung DA	Die Klasse Kategorie kann alle Attribute einer Kategorie fassen. In einem Array speichert das jeweilige Objekt die Unterkategorien der (Haupt-)Kategorie. Es gibt die drei Funktionen get, put und anzahl für den Unterkategorienarray.
kategorie_def.php	Unterkategorie <i>extends Kategorie</i>
Beschreibung DA	Unterkategorie ist eine Child-Klasse einer Kategorie. Sie übernimmt somit alle Attribute und Methoden. Die Unterkategorie erweitert die Kategorie um die Variable \$Unterkategorie_von. Darin wird jeweils der Name der (Haupt-)Kategorie gespeichert. Diese einfache Klassenbeziehung erlaubt (eigentlich) eine beliebig tiefe Verschachtelung von Kategorien.
kreditkarte_def.php	Kreditkarte

Modul: Klassendefinitionen	
Beschreibung DA	Definiert die Klasse Kreditkarte. Sie speichert für jede Kreditkarte einzeln den Namen des Kreditkarten-Instituts (Hersteller), das Handling und das benutzen-Flag. Das Handling kann von der Datenbank her nur zwei Werte annehmen: 'intern', 'extern'. Mit intern ist gemeint, dass die Bestellung komplett PhPepperShop intern abgewickelt wird. Wenn 'extern' angewählt wurde, kann man ein Pay-Objekt mit der kompletten Bestellung und Kunden-Information in der Datei payment_interface.php abholen und die Bestellung als akzeptiert oder nicht_akzeptiert wieder zurück geben. Man kann die eigentliche Zahlung dann in einem externen Programm (auf einem externen Server) abwickeln.
kunde_def.php	Kunde
Beschreibung DA	Definiert die Klasse Kunde, welche alle Daten eines Kunden fassen kann. Diese Klasse wurde vor allem dafür ausgelegt, alle möglichen Daten eines Kunden speichern zu können. Wenn man nicht alle Variablen benutzt, macht das aber auch nichts. Es sollen hiermit einfach alle Möglichkeiten abgedeckt sein.
versandkosten_def.php	Versandkosten
Beschreibung DA	Diese Klasse enthält alle von der Versandkostenberechnung her benötigten Variablen und einen Array mit den Versandkostenpreisen (z.B. ab 10kg CHF 10.00,...).
versandkosten_def.php	Versandkostenpreis
Beschreibung DA	In einer Versandkostenpreis-Klasse können alle Attribute der Tabelle versandkostenpreise gespeichert werden. Man speichert pro Versandkostenpreis-Objekt ein Preisintervall (von 10kg bis 20kg kosten die Versandkosten z.B. CHF 10.00).

Modul: Datenbank Wrapper	
Datei	Klassenname
database.php	TRecordSet / TSybaseRecordSet / TMySQLRecordSet
Beschreibung	TRecordSet ist eine abstrakte Klasse, welche für die jeweilige Datenbank-Implementierung vererbt wird. Ein RecordSet bietet Funktionen, um komfortabel und Datenbank unabhängig die einzelnen Felder auszulesen, die nächste Zeile einzulesen und die Anzahl erhaltener Zeilen auszugeben. (Funktionen: GetField(), NextRow())
database.php	TDatabase / TSybaseDatabase / TMySQLDatabase
Beschreibung	TDatabase ist ebenfalls eine abstrakte Klasse, welche für die jeweilige Datenbank-Implementierung weiter vererbt wird. Sie kümmert sich darum, datenbankunabhängige Connections zu bieten. Dies beinhaltet ein Connection-Objekt, Zugriffsfunktionen für Queries und eine vereinfachte Zugriffsfunktion ohne Rückgabewerte (INSERT, UPDATE). Funktionen: Exec(SQL-Query), Query(SQL-Query, welche ein Resultat liefert).

Modul: Datenbankanbindung	
Datei	
initialize.php	
Beschreibung DA	Wird von User-Modulen included und stellt ein Objekt \$Database zur Verfügung. Dieses Objekt eröffnet eine Verbindung als Shopuser. Die Verbindung ist nicht persistent und wird bei Script-Ende automatisch wieder geschlossen. Da das Login für jeden Shop wieder anders aussieht, erstellt das Installationsscript personalisierte Versionen dieser Datei aus einem Template.

Modul: Datenbankbindung	
ADMIN_initialize.php	
Beschreibung DA	Diese Datei wird von Administrator-Modulen eingebunden (included) und bietet eine nicht persistente Datenbank-Connection als Shopadmin an. Auch diese Datei wird aus einem Template erzeugt.

Modul: SQL-Statements	
Datei	
USER_SQL_BEFEHLE.php	
Beschreibung DA	Hier werden Variablen User-SQL-Queries zugewiesen (siehe Kapitel 2.4.7).
ADMIN_SQL_BEFEHLE.php	
Beschreibung DA	Hier werden wie bei USER_SQL_BEFEHLE.php die Query-Teile verschiedenen Variablen zugeordnet. Bloss handelt es sich hier um Queries von den Administrator-Funktionen. Diese Datei wird deshalb von USER-Modulen nicht included.

Modul: PHP-Funktionen - Artikel Handling	
Datei	Funktionsname
USER_ARTIKEL_HANDLING.php	get_Artikel
Beschreibung DA	Gibt auf Grund einer Artikel_ID ein Artikel-Objekt zurück. Argumente: Artikel_ID (INT) Rückgabewert: Ein Artikel (als Artikel-Objekt, siehe artikel_def.php)
USER_ARTIKEL_HANDLING.php	getArtikeleinerKategorie
Beschreibung DA	Alle Artikel einer Kategorie holen. Die Funktion getArtikeleinerKategorie liefert in einem Array alle Artikel, die einer Kategorie untergeordnet sind. Argumente: Kategorienname (String), Name der Parent-Kategorie, falls es eine Unterkategorie ist (String) Rückgabewert: Array (Key = Artikel_ID, Wert = Artikel-Objekt)
USER_ARTIKEL_HANDLING.php	IDgetArtikeleinerKategorie
Beschreibung DA	Alle Artikel einer Kategorie holen, als Argument wird eine Kategorie_ID erwartet. Die Funktion IDgetArtikeleinerKategorie liefert in einem Array alle Artikel die einer Kategorie untergeordnet sind. Argumente: Kategorie_ID (INT) Rückgabewert: Array (Key = Artikel_ID, Wert = Artikel-Objekt)
USER_ARTIKEL_HANDLING.php	getgesuchterArtikel

Modul: PHP-Funktionen - Artikel Handling	
<p>Beschreibung</p> <p>DA</p>	<p>Liefert aufgrund eines Such-Strings, der entweder im Namen oder in der Beschreibung des Artikels vorkommen kann, alle zutreffenden Artikel zurück. Es können jetzt auch mehrere Begriffe eingegeben werden. Diese werden dann konjunktiv verknüpft (AND). Auch Bilder werden optional angezeigt. Die beiden Argumente \$low- und \$highlimit definieren das Anzeigefenster: \$low sagt von wo, \$high sagt wieviele Artikel auf einmal ab \$low ausgelesen werden sollen.</p> <p>Argumente: Artikel_ID (INT), \$lowlimit (INT), \$highlimit (INT)</p> <p>Rückgabewert: Array (Key: Artikel-Objekt, Wert: Kategorienarray)</p>
USER_ARTIKEL_HANDLING.php	getallKategorien
<p>Beschreibung</p> <p>DA</p>	<p>Liefert ein Array mit allen Kategorien als Kategorie-Objekte zurück</p> <p>Argumente: keine</p> <p>Rückgabewert: Array mit Kategorien, welche ihre Unterkategorien enthalten</p>
USER_ARTIKEL_HANDLING.php	getKategorie_eines_Artikels
<p>Beschreibung</p> <p>DA</p>	<p>Liefert den Kategorienamen eines gegebenen Artikels. (Wenn die Kategorie KEINE Unterkategorie ist, so ist der Parent-Kategorie-Name gleich dem Kategorienamen).</p> <p>Argumente: Artikel_ID (INT)</p> <p>Rückgabewert: Array: Key = Kategorienname (String), Wert = Parent-Kategorienname (String)</p>
	getKategorieID_eines_Artikels
<p>Beschreibung</p> <p>DA</p>	<p>Liefert die Kategorie_IDs der Kategorien, in welchen der angegebene Artikel eingeteilt ist.</p> <p>Argumente: Artikel_ID (INT)</p> <p>Rückgabewert: Array: Kategorie_IDs</p>
USER_ARTIKEL_HANDLING.php	getKategorie
<p>Beschreibung</p> <p>DA</p>	<p>Liefert als Objekt eine (Unter-)Kategorie.</p> <p>Argumente: Kategorie_ID (INT)</p> <p>Rückgabewert: Ein Unterkategorie-Objekt</p>
USER_ARTIKEL_HANDLING.php	getvar_opt_preise
<p>Beschreibung</p>	<p>Liefert die Texte und Preise aller Optionen und Variationen eines Artikels. Diese Funktion wird benötigt, da nie Preisinformationen von Seite zu Seite übertragen werden (Sicherheit!). Preisinformationen jeglicher Art, kommen immer direkt von der Datenbank in die Ziel-Page.</p> <p>Argumente: Artikel_ID (INT)</p> <p>Rückgabewert: Ein Artikel-Objekt, das Optionen und Variationen enthält</p>

Modul: PHP-Funktionen - Userseitiges Bestellwesen	
Datei	Funktionsname
USER_BESTELLUNG.php	getBestellung

Modul: PHP-Funktionen - Userseitiges Bestellwesen	
Beschreibung DA	Gibt aufgrund einer Session-ID (falls existierend) eine Bestellung als Bestel- lungs-Objekt zurück (inkl. allen zugeordneten Artikeln). Argumente: Session_ID (String) Rückgabewert: Bestelungs-Objekt
USER_BESTELLUNG.php	test_create_Bestellung
Beschreibung DA	Diese Funktion gehört einerseits zum Bestellwesen, andererseits aber auch zum Session-Management. Sie überprüft zuerst, ob in der Datenbank eine zur aktuellen Session-ID pas- sende Bestellung existiert. Wenn ja, ist alles gut, wenn nein, so wird eine Neue erstellt. Auf diese Weise ist sichergestellt, dass jeder Benutzer jederzeit ohne Fehlermeldung in den Bestellvorgang einsteigen kann. Argumente: Session_ID (String) Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	addArtikel
Beschreibung	Diese Funktion wird benötigt, wenn der Kunde auf den "In den Warenkorb"- Button klickt. Sie fügt der Bestellung des Kunden den Artikel hinzu (Zuord- nung). Argumente: - Session_ID (String), damit man weiss welcher Bestellung der Artikel zuzuordnen ist - Artikel_info-Objekt, enthält alle für die Bestellung relevanten Felder eines Artikels (inkl. Anzahl u.s.w.) Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	del_B_Artikel
Beschreibung DA	Löscht einen Artikel-Eintrag für eine Bestellung. (Komische Namensgebung, weil es delArtikel schon gibt) Argumente: Artikel_ID (INT) und Bestelungs_ID (INT) um die betroffenen Artikel eindeutig zu bestimmen Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	delSession
Beschreibung DA	Löscht eine Session und damit auch die Bestellung. Es werden drei Schritte ab- gearbeitet: 1.) Bestellung löschen 2.) Artikelliste der Bestellung löschen 3.) Kunde-Bestellung-Referenz löschen (Tabelle bestellung_kunde) Argumente: - \$Session_ID (String) wird benötigt um die Bestellung in der Ta- belle 'bestellung' zu referenzieren. - \$Bestelungs_ID (INT) wird benötigt, damit man OHNE eine weitere Query zu starten auch alle zur gelöschten Bestellung gehörenden Artikel in der artikel_bestellung-Tabelle löschen kann) Rückgabewert: true bei Erfolg, sonst Abbruch per die-Funktion
USER_BESTELLUNG.php	delallexpiredSessions

Modul: PHP-Funktionen - Userseitiges Bestellwesens	
Beschreibung DA	Löscht ALLE abgelaufenen Bestellungen und ihre Artikel- und Kunden-Referenzen. Problematik: Dies ist eine USER-Funktion, die aber eine Administrator-Operation ausführt --> Sicherheitslücke(!?). Besser wäre eine Art Cron-Job, der mit Admin-Rechten diese Operation ausführt. Wir dachten uns aber, auf diese Weise muss der Admin nichts machen und der Shop hält die Anzahl der eingetragenen Bestellungen (bei ausgeschaltetem Bestellmanagement) von alleine in Grenzen. Diese Funktion ist zeitaufwändig und soll an einem selten besuchten Platz aufgerufen werden (bei uns nach Beststellungs-Abschluss). Argumente: keine Rückgabewert: keiner
USER_BESTELLUNG.php	getversandkostensettings
Beschreibung DA	Liefert die Versandkosten-Einstellungen des Shops Argumente: Setting_Nr (im Hinblick auf mehrere zu verwaltende Shops) Rückgabewert: Versandkosten-Objekt
USER_BESTELLUNG.php	getversandkostentext
Beschreibung DA	Liefert den Beschreibungstext des Rechnungspostens für die Versandkosten im Warenkorb. (z.B. "Versand- und Verpackungskosten") Argumente: Setting_Nr (im Hinblick auf mehrere zu verwaltende Shops) Rückgabewert: String
USER_BESTELLUNG.php	berechneversandkosten
Beschreibung DA	Berechnet die Versandkosten einer Bestellung und schreibt die Werte auch gleich in die entsprechenden Variablen der angegebenen Bestellung. Es wird je nach Zahlungsart noch die Nachnahmegebühr und ein Mindermengenzuschlag hinzugefügt. Argumente: Session_ID (String) Rückgabewert: Array: <ul style="list-style-type: none"> 1.Element = Versandkosten 2.Element = Mindermengenzuschlag 3.Element = Rechnungstotal (ALLES inkl. allem) 4.Element = Nachnahmegebühr
USER_BESTELLUNG.php	getKunde
Beschreibung DA	Gibt aufgrund einer Kunden_ID ein Kunden-Objekt zurück. Argumente: Kunden_ID (String) Rückgabewert: Ein Kunde in Form eines Kunden-Objekts (siehe kunde_def.php)
USER_BESTELLUNG.php	getallKunden
Beschreibung DA	Gibt in einem Array von Kunden-Objekten alle Kunden zurück. Argumente: keine Rückgabewert: Array mit Kunden-Objekten (siehe kunde_def.php)
USER_BESTELLUNG.php	newKunde

Modul: PHP-Funktionen - Userseitiges Bestellwesens	
Beschreibung DA	Fügt der Datenbank einen neuen Kunden hinzu (betroffene Tabellen kunde, bestellung_kunde). Dabei wird auch eine neue Kunden_ID berechnet, siehe Kapitel Security, Kunden-ID Kapitel 2.1.6. Argumente: Ein Kunde in 'Einzelteilen': (vom Formular her) \$Kunden_ID,\$Session_ID,\$Kunden_Nr,\$Anrede,\$Vorname,\$Nachname,\$Firma,\$Abteilung,\$Strasse,\$Postfach,\$PLZ,\$Ort,\$Land,\$Tel,\$Fax,\$Email,\$Einkaufsvolumen,\$Login,\$Passwort,\$gesperrt,\$temp,\$Attribut1,\$Attribut2,\$Attribut3,\$Attribut4,\$Attributwert1,\$Attributwert2,\$Attributwert3,\$Attributwert4 Rückgabewert: Die Kunden_ID (String) des neuen Kunden
USER_BESTELLUNG.php	delKunde
Beschreibung DA	Löscht einen Kunden in der Datenbank. Argumente: Kunden_ID (String) Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	getKunde_einer_Bestellung
Beschreibung DA	Gibt aufgrund einer Bestellungen_ID ein Kunden-Objekt zurück. Argumente: Bestellungen_ID (INT) Rückgabewert: Kunden_Objekt (siehe kunde_def.php)
USER_BESTELLUNG.php	updKunde
Beschreibung DA	Dies ist ein Update aller Kundendaten. (Diese Funktion wird momentan nicht verwendet, erlaubt aber das Setzen ALLER Kunden-Attribute) Argumente: Kunde in Einzel-Attributen: (von einem Formular her) \$Kunden_Nr,\$Session_ID,\$Anrede,\$Vorname,\$Nachname,\$Firma,\$Abteilung,\$Strasse,\$Postfach,\$PLZ,\$Ort,\$Land,\$Tel,\$Fax,\$Email,\$Einkaufsvolumen,\$Login,\$Passwort,\$gesperrt,\$temp,\$Attribut1,\$Attribut2,\$Attribut3,\$Attribut4,\$Attributwert1,\$Attributwert2,\$Attributwert3,\$Attributwert4,\$Kunden_ID Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	updKundenFelder
Beschreibung DA	Dies ist ein Update aller vom Kunden beeinflussbaren / benötigten Felder seines Datensatzes. Argumente: Kundendaten in Einzelteilen, vom Formular her: \$Session_ID,\$Anrede,\$Vorname,\$Nachname,\$Firma,\$Abteilung,\$Strasse,\$Postfach,\$PLZ,\$Ort,\$Land,\$Tel,\$Fax,\$Email,\$Attribut1,\$Attribut2,\$Attribut3,\$Attribut4,\$Attributwert1,\$Attributwert2,\$Attributwert3,\$Attributwert4,\$Kunden_ID Rückgabewert: true bei Erfolg, sonst Abbruch per die-Funktion
USER_BESTELLUNG.php	gibBestellung_an_Kunde
Beschreibung DA	Ordnet eine Bestellung einem Kunden zu (in Tabelle bestellung_kunde). Argumente: Bestellungen_ID (INT), Kunden_ID (String) Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	delBestellung_von_Kunde

Modul: PHP-Funktionen - Userseitiges Bestellwesens	
Beschreibung DA	Löscht alle Bestellungen eines Kunden. Wird benötigt um persistente Kunden, ihre Bestellungen und deren Referenzen zum zu löschenden Kunden zu entfernen. Zuerst werden alle zu löschenden Bestellungen_IDs ausgelesen, dann alle Referenzen vom Kunden zu den Bestellungen gelöscht und zum Schluss die Bestellungen entfernt. Argumente: Kunden_ID (String) Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	getAttributobjekt
Beschreibung DA	Gibt ein Attribut-Objekt zurück (darin befinden sich in Arrays alle Attribute, sortiert nach ihrer jeweiligen Positions-Nr). Argumente: keine Rückgabewert: Attribut-Objekt
USER_BESTELLUNG.php	setAttributobjekt
Beschreibung DA	Setzt alle Attribute in der Tabelle attribut. Argumente: Attribut-Objekt Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	checkLogin
Beschreibung DA	Diese Funktion überprüft die eingegebenen Login-Daten und veranlasst dann, je nach Eingabe: - einen neuen temporären Kunden anzulegen - die Kunden_ID eines persistenten Kunden auszulesen und seine Session_ID upzudaten Argumente: Login (String), Passwort (String), Session_ID (String) Rückgabewert: - User existiert, Passwort stimmt --> existierende Kunden_ID (String) - User existiert, Passwort stimmt nicht --> "P" (String) - User existiert nicht --> neue Kunden_ID (String)
USER_BESTELLUNG.php	checkSession
Beschreibung DA	Diese Funktion gibt die Kunden_ID zurück, falls ein Kunde eingeloggt ist und sich schon authentifiziert hat. Argumente: Session_ID (String) Rückgabewert: User existiert und ist eingeloggt --> Kunden_ID (String) User ist nicht eingeloggt --> "" (Leer-String)
USER_BESTELLUNG.php	mailPasswort

Modul: PHP-Funktionen - Userseitiges Bestellweszen	
Beschreibung DA	Diese Funktion sendet dem Benutzer, der sein Passwort vergessen hat, sein Passwort an seine im System gespeicherte E-Mail Adresse. Zur 'Verifikation' muss er sein Login eingeben. (Zuordnung zur E-Mail-Adresse). Argumente: Login (String) Rückgabewert: true bei Erfolg false wenn kein Login gefunden wurde sonst Abbruch per die-Funktion
USER_BESTELLUNG.php	addEinkaufsvolumen
Beschreibung DA	Diese Funktion addiert das Bestellungstotal zum Einkaufsvolumen eines Kunden, ebenso wird das LetzteBestellung-Attribut erneuert. Argumente: Session_ID (String), Betrag (Double) Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	updBestellungsFelder
Beschreibung DA	Hier werden die zur Bestellung gehörenden Daten eines Kunden zwischengespeichert (Anmerkung, Datum, Zahlungsart, Kreditkartendaten, Zusatzattribute1 bis 4 und ihre Namen) Argumente: Session_ID (String) (Damit man weiss, zu welcher Bestellung die Daten gehören werden. Bestellungen-Objekt enthält die User-Daten (Def. siehe bestellung_def.php)) Rückgabewert: true bei Erfolg
USER_BESTELLUNG.php	getEmailMessage
Beschreibung DA	Da bei der Kreditkartenzahlung mit externer Zahlungsabwicklung der E-Mail-Message String verloren gehen würde, wird dieser temporär in der jeweiligen Bestellung zwischengespeichert. Diese Funktion liest den temporär zwischengespeicherten E-Mail-Message-String aus und LÖSCHT den E-Mail-Message-String in der entsprechenden Bestellung! Argumente: Session_ID (String) Rückgabewert: String oder Abbruch per die-Funktion
USER_BESTELLUNG.php	putEmailMessage
Beschreibung DA	Da bei der Kreditkartenzahlung mit externer Zahlungsabwicklung der E-Mail-Message String verloren gehen würde, wird dieser temporär in der jeweiligen Bestellung zwischengespeichert. Mit dieser Funktion kann man den E-Mail-Message-String temporär der Bestellung des Kunden zuweisen. Argumente: Emailstring (String), Session_ID (String) Rückgabewert: true bei Erfolg oder Abbruch per die-Funktion
USER_BESTELLUNG.php	validateCC
Beschreibung DA	Die folgende Funktion von <i>Error Brett</i> (Brett@InterWebDesign.com) überprüft die Kreditkartennummer auf Vollständigkeit und Prüfsumme (MOD-10 Verfahren). Das Datum wird nicht hier validiert. Argumente: Kreditkartennummer (String), Kreditkarten_Institut (String) Rückgabewert: - true bei korrekter Kreditkartennummer - false bei falscher Kreditkartennummer - (-1) bei nicht korrektem Kreditkarten Institut
USER_BESTELLUNG.php	toCharArray

Modul: PHP-Funktionen - Userseitiges Bestellweszen	
Beschreibung DA	Diese Funktion gibt eine Zeichenkette als Array zurück. Sie wird bis jetzt nur von validateCC benötigt. Argumente: String Rückgabewert: Array der Zeichen vom Eingabestring
USER_BESTELLUNG.php	schliessenBestellung
Beschreibung DA	Damit ein Kunde seine Bestellung nicht mehr abändern kann, wird seine Bestellung mit dem Attribut <code>Bestellung_abgeschlossen</code> in der Tabelle <code>bestellung</code> als unveränderbar markiert. In der Datenbank wird ferner die Session-ID bei diesem Eintrag gelöscht. Somit erhält er mit der gleichen Session-ID eine neue Bestellung in der Datenbank (siehe auch <code>test_create_Session</code>). Argumente: <code>Session_ID</code> (String), <code>abschliessen_Flag</code> enum('Y','N') Rückgabewert: true bei Erfolg, sonst Abbruch per die-Funktion

Modul: PHP-Funktionen – Einzelne Shop-Settings auslesen	
Datei	Funktionsname
USER_ARTIKEL_HANDLING.php	getWaehrung
Beschreibung	Liefert die Währung in einem String zurück. (z.B. SFr., CHF, DM, ATS) Argumente: keine Rückgabewert: Währung (String, max. 4 Zeichen)
USER_ARTIKEL_HANDLING.php	getGewichts_Masseinheit
Beschreibung	Liefert die Gewichts-Masseinheit in einem String zurück. (z.B. kg, ml, g, t,...) Argumente: keine Rückgabewert: Gewichts-Masseinheit (String)
USER_ARTIKEL_HANDLING.php	getThumbnail_Breite
Beschreibung	Da in <code>bild_up.php</code> ein Thumbnail eines Artikel-Bildes erzeugt wird, kann über dieses <code>shop_settings</code> -Attribut die Breite des Thumbnails festgelegt werden. Diese Funktion liefert nun den Integer-Wert in Pixel. Argumente: keine Rückgabewert: Thumbnail-Breite (INT)
USER_ARTIKEL_HANDLING.php	getshopname
Beschreibung	Liefert den in den Shop-Settings gespeicherten Namen des Shops in einem String. Argumente: keine Rückgabewert: Shopname (String)
USER_ARTIKEL_HANDLING.php	getmax_session_time
Beschreibung	In den Shop-Settings lässt sich ein fixer Wert in Sekunden einstellen, der als die maximal gültige Dauer einer Session zählt. So ist man etwas freier bei der Wahl der Session-Dauer. (Nach oben gibt es die Limite in der <code>php.ini</code>). Diese Funktion liefert nun diesen Wert als Integer zurück. Argumente: keine Rückgabewert: Maximale Dauer einer User-Session (INT)

Modul: PHP-Funktionen – Einzelne Shop-Settings auslesen	
USER_ARTIKEL_HANDLING.php	getBezahlungsart
Beschreibung	Liefert einen Array, in dem steht, welche Zahlungsarten gewählt werden können. Der jeweilige Name ist 'Y', wenn er in den Shop-Settings aktiviert wurde, sonst 'N'. Argumente: keine Rückgabewert: Array (Key = Zahlungsart, Wert = 'Y' oder 'N')
USER_ARTIKEL_HANDLING.php	getAGB
Beschreibung	Liefert die Allgemeinen Geschäftsbedingungen (AGB) aus der Datenbank als String (Tabelle shop_settings). Argumente: keine Rückgabewert: Allgemeine Geschäftsbedingungen (String)
USER_ARTIKEL_HANDLING.php	getEmail
Beschreibung	Liefert die E-Mail Adresse als String passend zur Bestellung, welche von der im Argument angegebenen Session_ID referenziert wird (Kunden E-Mail Adresse). Argumente: Session_ID (String) Rückgabewert: E-Mail-Adresse als String (Eingaben ungeprüft!)
USER_ARTIKEL_HANDLING.php	getSuchInkrement
Beschreibung DA	Liefert das Such-Inkrement als Integer --> Wieviele gefundene Resultate auf einmal anzeigen (siehe Artikelsuche) Argumente: keine Rückgabewert: Such-Inkrement (INT)
USER_ARTIKEL_HANDLING.php	getBestellungsmanagement
Beschreibung DA	Liefert 'Y' oder 'N', je nachdem ob das Bestellungsmanagement erwünscht ist oder nicht. Eingestellt wird dieses Attribut in den allgemeinen Shop Einstellungen (setshopsettings). Argumente: keine Rückgabewert: 'Y' oder 'N' (String)
USER_ARTIKEL_HANDLING.php	getvaroptinc
Beschreibung DA	Liest aus den Shop-Settings aus, wieviele Felder für die Erfassung von Optionen und Variationen bei einem Artikel mindestens dargestellt werden. Zusätzlich wird noch der Wert ausgelesen, wie viele leere Felder angezeigt werden, wenn die Anzahl der eingegeben Optionen / Variationen die Anzahl der eingestellten Minimalfelderanzahl übersteigt. Argumente: keine Rückgabewert: <i>entweder</i> Array: 1. Element = Optionen Inkrement (Opt_inc), 2. Element = Variations Inkrement (Var_inc) 3. Element = Optionen Basisanzahl (Opt_anz) 4. Element = Variations Basisanzahl (Var_anz) <i>oder</i> Abbruch per die-Funktion

Modul: PHP-Funktionen – Einzelne Shop-Settings auslesen	
USER_ARTIKEL_HANDLING.php	getSSL
Beschreibung DA	<p>Diese Funktion überprüft, ob SSL eingeschaltet wurde. Wenn ja, wird eine absolute URL zusammengebaut und https vorangestellt. Um wieder aus dem SSL-Bereich austreten zu können, kann man der Funktion im Flag \$noSSL (true) mitteilen, dass man den SSL-Bereich verlassen will. Es wird dann als Rückgabewert eine URL mit http zurückgegeben. Mit der Variable \$server wird meistens \$SERVER_NAME übergeben.</p> <p>Ein Aufruf dieser Funktion sieht typischerweise so aus: getSSL(\$PHP_SELF, \$SERVER_NAME, false).</p> <p>Eingestellt wird SSL in den allgemeinen Shop Einstellungen (setshopsettings)</p> <p>Argumente: \$oldphpself (String, \$PHP_SELF), \$server (String, \$SERVER_NAME), \$noSSL (boolean)</p> <p>Rückgabewert: Neuer URL (mit oder ohne https) (String)</p>
USER_ARTIKEL_HANDLING.php	getNachnahmebetrag
Beschreibung DA	<p>Liefert den Geldwert, welcher bei einer Nachnahmelieferung zu verrechnen ist (Nachnahmegebühr).</p> <p>Argumente: keine</p> <p>Rückgabewert: Nachnahmebetrag (DOUBLE), 0.0 = keiner vorhanden</p>
USER_ARTIKEL_HANDLING.php	getKontoinformation
Beschreibung DA	<p>Liefert den String, welchen man in den allgemeinen Shop-Einstellungen neben Vorkasse im Feld Kontoinformation eingeben kann.</p> <p>Argumente: keine</p> <p>Rückgabewert: Kontoinformation (String)</p>
USER_ARTIKEL_HANDLING.php	getcassarg
Beschreibung DA	<p>Einen CSS-String aus der Tabelle css_file auslesen:</p> <p>Argumente: CSS-ID</p> <p>Rückgabewert: zugehöriger CSS-String</p>
USER_ARTIKEL_HANDLING.php	getKreditkarten
Beschreibung DA	<p>Diese Funktion liest alle Kreditkartendaten aus der kreditkarten-Tabelle aus und gibt alle Kreditkarten-Objekte in einem Array zurück.</p> <p>Argumente: keine</p> <p>Rückgabewert: Array von Kreditkarten-Objekten</p>
USER_ARTIKEL_HANDLING.php	getShopEmail
Beschreibung	<p>Liefert die E-Mail Adresse des Shop Betreibers (oder des Verantwortlichen --> in Shop-Settings eingestellt)</p> <p>Argumente: keine</p> <p>Rückgabewert: Shop-Email-Adresse (String)</p>

Modul: PHP-Funktionen – Administrationsfunktionen	
Datei	Funktionsname
SHOP_ADMINISTRATION.php	newArtikel

Modul: PHP-Funktionen – Administrationsfunktionen	
Beschreibung	Einen neuen Artikel in die Datenbank speichern.
DA	<p>Diese Funktion fügt einen neuen Artikel ohne Bild in die Datenbank ein.</p> <p>Sie wird von der Datei bild_up.php aufgerufen. (Betrifft Tabellen artikel, artikel_optionen, artikel_variationen, artikel_kategorien). Damit mehrere Kategorien pro Artikel möglich sind, werden die Kategorie_IDs in einem Array übergeben. Eine weitere Spezialität ist das Flag \$kat_save. Nur wenn dieses = true (1) ist, wird der neue Artikel auch in die angegebenen Kategorien eingetragen (siehe auch updArtikel_2)</p> <p>Argumente: In Einzelteile zerlegter Artikel, vom Formular her: \$Kategorie_IDarray,\$Artikel_Nr,\$Name,\$Beschreibung,\$letzteAenderung,\$Preis,\$Aktionspreis,\$Gewicht,\$Link,\$Optionenarray,\$Variationsarray,\$kat_save</p> <p>Rückgabewert: Artikel_ID (INT) des neu eingefügten Artikels. (Artikel_ID wurde per Auto-Increment Funktion erzeugt (MySQL). Hier sei noch erwähnt, dass der DB-Wrapper (database.php) nur für MySQL diese Funktionalität bietet. Bei den anderen Datenbanken, auch Sybase, muss das noch nachprogrammiert werden.</p>
SHOP_ADMINISTRATION.php	delArtikel
Beschreibung	Einen Artikel aus der Shop-Datenbank entfernen (Betrifft mehrere Tabellen).
DA	<p>Argumente: Artikel_ID (INT)</p> <p>Rückgabewert: 1 (entspricht true) bei Erfolg</p>
SHOP_ADMINISTRATION.php	updArtikel
Beschreibung	<p>Einen schon bestehenden Artikel in der Datenbank verändern (updaten):</p> <p>Diese Funktion ist etwas verzettelt. Sie wurde seit der Version 1.05 komplett überarbeitet und zum grossen Teil neu programmiert.</p> <p>Auf die Schnelle erklärt, wird ein Artikel wie folgt upgedated:</p> <ol style="list-style-type: none"> 1.) Daten des zu updatenden Artikels einlesen 2.) Per darstellenArtikel-Funktion (SHOP_ADMINISTRATION_ARTIKEL.php) diesen im Eingabeformular darstellen [3.] Optionaler Bild-Einfüge-Dialog] (bild_up.php, hier drin geht es auch weiter mit dem Update) 4.) Über die updArtikel_2 Funktion den Artikel updaten (siehe gleich unter dieser Funktion) <p>Argumente: Artikel_ID (INT)</p> <p>Rückgabewert:</p> <ul style="list-style-type: none"> - Artikel_ID bei Erfolg - false, bei erfolgreichem Artikel Update- der Artikel wurde in die Kategorie Nichtzugeordnet abgelegt - Abbruch der Funktion per die-Funktion und entsprechender Fehlermeldung
SHOP_ADMINISTRATION.php	updArtikel_2

Modul: PHP-Funktionen – Administrationsfunktionen	
Beschreibung DA	<p>Diese Funktion wird von der Datei bild_up.php aufgerufen, sobald ein Artikel upgedated werden soll. Dies ist quasi der zweite Teil eines Artikel Updates. Diese Funktion updated nur die Artikeldaten OHNE das Bild (Weiteres zum eigentlichen Bild, siehe bild_up.php).</p> <p>Das Flag \$kat_save wird benutzt um den Artikel upzudaten, aber nicht um die Kategorien-Referenzen des Artikels zu aktualisieren (MEHR-OPTIONEN / VARIATIONEN Button Implementation). Es bewirkt, dass Teil 4 von updArtikel_2 übersprungen wird.</p> <p>Argumente: "In Einzelteile zerlegter Artikel, vom Formular her" \$Kategorie_IDarray,\$Artikel_ID,\$Artikel_Nr,\$Name,\$Beschreibung,\$letzteAenderung,\$Preis,\$Aktionspreis,\$Gewicht,\$Link,\$Optionenarray,\$Variationsarray,\$kat_save</p> <p>Rückgabewert: Artikel_ID (INT) bei Erfolg</p>
SHOP_ADMINISTRATION.php	movengzArtikel
Beschreibung DA	<p>Kopiert den angegebenen Artikel in die Kategorie 'Nichtzugeordnet'.</p> <p>Argumente: Artikel_ID (INT)</p> <p>Rückgabewert: true bei Erfolg</p>
SHOP_ADMINISTRATION.php	getshopsettings
Beschreibung DA	<p>Liefert die in der Datenbank in der Tabelle shop_settings definierten Shop-Einstellungen als einen Array of String.</p> <p>Argumente: keine</p> <p>Rückgabewert: Array: Keys = Feldnamen, Werte = Werte der Felder</p>
SHOP_ADMINISTRATION.php	setshopsettings
Beschreibung DA	<p>Macht ein Update auf alle Attribute der Tabelle shop_settings und aktualisiert somit alle Shop-Settings.</p> <p>Argumente: Alle Shopsettings einzeln (dies ist praktisch, da diese Funktion nur nach einem Formular aufgerufen wird): \$MwStsatz,\$MwStpflichtig,\$MwStNummer,\$Admin_pwd,\$Name,\$Adresse1,\$Adresse2,\$PLZOrt,\$Tel1,\$Tel2,\$Email,\$Thumbnail_Breite,\$Mindermengenzuschlag_Aufpreis,\$Abrechnung_nach_Preis,\$Abrechnung_nach_Gewicht,\$Mindermengenzuschlag,\$Kreditkarten_Postcard,\$ShopVersion,\$Abrechnung_nach_Pauschale,\$Rechnung,\$ShopVersion,\$Waehrung,\$Nachnahme,\$Mindermengenzuschlag_bis_Preis,\$keineVersandkostenmehr_ab,\$keineVersandkostenmehr,\$SSL,\$Bestellungsmanagement,\$Gewichts_Masseinheit,\$max_session_time,\$AGB,\$Opt_inc,\$Var_inc,\$Opt_anz,\$Var_anz,\$SuchInkrement,\$Vorauskasse,\$Kontoinformation</p> <p>Rückgabewert: true bei Erfolg</p>
SHOP_ADMINISTRATION.php	delBild
Beschreibung	<p>Löscht das Bild des aktuellen Artikels.</p> <p>Löscht das Bild eines Artikels in der DB (Wenn man dem Artikel kein Bild mehr zuordnen will. Diese Funktion wird beim Update eines Artikels verwendet).</p> <p>Argumente: Artikel_ID (INT)</p> <p>Rückgabewert: true bei Erfolg</p>
SHOP_ADMINISTRATION.php	newKategorie

Modul: PHP-Funktionen – Administrationsfunktionen	
Beschreibung DA	Fügt eine neue Kategorie in die Tabelle 'kategorien' ein. Unterstützt vorerst NUR den Namen und die Positionsnummer. Es ist noch keine Beschreibung und Bild-Eingabe möglich. Seit v.1.05 werden auch Unterkategorien unterstützt. Argumente: Name (String), Positions-Nr (INT), Unterkategorie_von (String) der neuen Kategorie / Unterkategorie. Rückgabewert: true bei Erfolg
SHOP_ADMINISTRATION.php	delKategorie
Beschreibung DA	Eine Kategorie löschen: Dabei werden die noch darin enthaltenen Artikel, sofern sie keiner weiteren, noch existierenden Kategorie angehören, in die Kategorie 'Nichtzugeordnet' abgelegt. Diese Funktion benötigt seit der Implementierung von Unterkategorien und Mehrfachkategorien pro Artikel eine komplexe Logik. Ist das Artikelloeschen_Flag gleich true, so werden die Artikel gelöscht. Argumente: Kategorie_ID (INT), Artikelloeschen_Flag (boolean) Rückgabewert: true bei Erfolg
SHOP_ADMINISTRATION.php	verschiebenKategorie
Beschreibung DA	Eine (Unter-)Kategorie verschieben. Die Funktionalität, um die Positionsnummern nach dem Verschieben wieder richtig zu verteilen, wurde in die Funktion katposschieben verlegt. Argumente: Kategorie_ID (INT), neue Positionsnummer (INT), aktuelle- und neue Unterkategorie (Strings) Rückgabewert: true bei Erfolg
SHOP_ADMINISTRATION.php	umbenennenKategorie
Beschreibung DA	Eine (Unter-)Kategorie umbenennen. Es wird der Name der entsprechenden Kategorie aktualisiert. Falls die Kategorie Unterkategorien besitzt, wird deren Attribut Unterkategorie_von nachgeführt. Argumente: Kategorie_ID (INT), Neuer_Name (String) Rückgabewert: true oder Abbruch per die-Funktion
SHOP_ADMINISTRATION.php	katposschieben
Beschreibung DA	Diese Funktion ist eine Hilfsfunktion im Kategorien-Management. Sie verschiebt die Positions-Nummern der Kategorien entsprechend den Eingabe-Parametern. Positions-Nummern ordnen die Kategorien-Anzeige aufsteigend ein. Nachdem diese Funktion abgelaufen ist, kann man (z.B.) entweder eine neue (Unter-)Kategorie mit gewünschter Pos_Nr eintragen (eine Lücke wurde geschaffen), oder es wurde eine Lücke gelöscht --> z.B. wie bei delKategorie. Argumente: Kategorie_ID (INT), \$currentPos (INT), \$newPos (INT), \$delPos (INT), \$currentUkat (String), \$newUkat (String) Rückgabewert: true oder Abbruch per die-Funktion
SHOP_ADMINISTRATION.php	getNichtzugeordnetKategorie

Modul: PHP-Funktionen – Administrationsfunktionen	
Beschreibung DA	Liefert als Kategorie-Objekt die spezielle Nichtzugeordnet Kategorie. Diese Kategorie wird erkannt, indem sie in ihrem Attribut Unterkategorie_von den String @PhPepperShop@ trägt. Diese Funktion wurde geschrieben, weil die Kategorie in einem englischen Shop wahrscheinlich nicht mehr 'Nichtzugeordnet' heissen wird. Auf diese Weise kann man diese Kategorie immer auslesen, egal wie sie heisst. Argumente: keine Rückgabewert: ein Kategorie-Objekt
SHOP_ADMINISTRATION.php	updatecssarg
Beschreibung DA	Einen CSS-String in der Tabelle css_file updaten. Argumente: CSS-ID (INT), CSS-String (String) Rückgabewert: true, sonst Funktionsabbruch per die
SHOP_ADMINISTRATION.php	setvaroptinc
Beschreibung DA	Die Inkremente für weitere leere Felder in der darstellen_Artikel-Funktion setzen (Tabelle shop_settings, Attribute: Opt_inc, Var_inc). Argumente: Optionsinkrement (INT), Variationsinkrement (INT) Rückgabewert: Entweder true (1) oder Abbruch per die-Funktion
SHOP_ADMINISTRATION.php	setversandkostensettings
Beschreibung DA	Speichert alle Versandkosten-Einstellungen in der Datenbank ab. Argumente: Man gibt alle Versandkostensettings einzeln an. Dies ist praktisch, da diese Funktion nur nach einem Formular aufgerufen wird: \$Abrechnung_nach_Preis,\$Abrechnung_nach_Gewicht,\$Abrechnung_nach_Pauschale,\$Pauschale_text,\$keineVersandkostenmehr,\$keineVersandkostenmehr_ab,\$Mindermengenzuschlag,\$Mindermengenzuschlag_bis_Preis,\$Mindermengenzuschlag_Aufpreis,\$Nachnamebetrag,\$Setting_Nr,\$Versandkostenpreise Rückgabewert: Die Funktion liefert true bei Erfolg
SHOP_ADMINISTRATION.php	getBestellung_Ref
Beschreibung DA	Diese Funktion liefert eine Bestellung aufgrund einer Referenz-Nr zurück. Eine Referenz_Nr ist die Summe aus dem Offset 154870 und der aktuellen Bestellungen_ID. Die Referenznummer wird dem Kunden auf seinem E-Mail angegeben und soll z.B. bei Anfragen das Suchen nach Bestellungen vereinfachen. Argument: Referenz_Nr (INT) Rückgabewert: Eine Bestellung als Bestellungen-Objekt
SHOP_ADMINISTRATION.php	getBestellung_Kunde
Beschreibung DA	Gibt auf Grund eines Kundennamens die dazugehörige(n) Bestellung(en) zurück. Argument: Name und Vorname des Kunden (Strings) Rückgabewert: Ein Array von Bestellung(-en) als Bestellungen-Objekt(-e) (Definition siehe bestellung_def.php)
SHOP_ADMINISTRATION.php	getBestellung_Alle

Modul: PHP-Funktionen – Administrationsfunktionen	
Beschreibung DA	Gibt alle als abgeschlossen markierten Bestellungen in einem Array zurück. Argumente: keine Rückgabewert: Ein Array von Bestellungen als Bestellungs-Objekte
SHOP_ADMINISTRATION.php	delBestellung
Beschreibung DA	Diese Funktion löscht eine abgeschlossene Bestellung unwiderruflich. Natürlich werden auch all ihre Referenzen in der Tabelle artikel_bestellung gelöscht. Als dritter Teil werden noch alle Referenzen zum Kunden der Bestellung gelöscht. Argumente: Bestellungen_ID (INT) Rückgabewert: true bei Erfolg
SHOP_ADMINISTRATION.php	setKreditkarten
Beschreibung DA	Diese Funktion schreibt alle Kreditkartendaten in die kreditkarten-Tabelle (Speichern der Kreditkarten-Institute und ihrer Optionen). Vorgehen: Vorandene Daten löschen, neue Daten speichern (Kategorie_ID inkrementiert) Argumente: Herstellerarray (Array of String), benutzenarray (Array of enum('Y','N')), Handlingarray (Array of enum('Y','N')) Rückgabewert: true bei Erfolg

Modul: PHP-Funktionen – Shop Layout	
Datei	Funktionsname
SHOP_LAYOUT.php	mkindexphp
Beschreibung DA	Erstellt die Datei index.php aus dem Template-File indextemplate.txt. Tags, welche im Templatefile in Doppelklammern (<< >>) stehen, werden durch die entsprechenden Einträge in der Datenbank (Tabelle: cssfile) ersetzt. Argumente: keine Rückgabewert: false, wenn ein Fehler aufgetreten ist
SHOP_LAYOUT.php	mkcssfiles
Beschreibung DA	Erstellt die CSS-Dateien (shopstyles.css) in den Verzeichnissen shop, shop/Frameset und shop/Admin aus dem Templatefile csstemplate.txt. Tags, welche im Templatefile in Doppelklammern (<< >>) stehen, werden durch die entsprechenden Einträge in der Datenbank (Tabelle: cssfile) ersetzt. Argumente: keine Rückgabewert: false, wenn ein Fehler aufgetreten ist
SHOP_LAYOUT.php	htmltext

Modul: PHP-Funktionen – Shop Layout	
<p>Beschreibung</p> <p>DA</p>	<p>Gibt eine Tabelle im HTML-Format aus, die folgende Elemente enthält:</p> <ul style="list-style-type: none"> - R,G,B Editierfelder für Schriftfarbe - dropdown-Box für Schriftgewicht - dropdown-Box für Schriftgröße - dropdown-Box für Text Decoration <p>Die Parameter werden aus der Datenbank ausgelesen und direkt in die angezeigten Felder eingefüllt</p> <p>Argumente:</p> <ul style="list-style-type: none"> \$id -> identifizier in datenbank \$label -> Benennung des Tags, z.B ShopTitel \$color -> wenn 1 (true) wird Farbauswahl angezeigt \$weight -> wenn 1 (true) wird Schriftgewicht angezeigt \$size -> wenn 1 (true) wird Schriftgröße angezeigt \$style -> wenn 1 (true) wird Text Style angezeigt <p>Rückgabewert: HTML Tabelle mit oben genannten Formularelementen.</p> <p>Formular Bezeichner bestehen aus der Variablen \$id und folgendem Text.:</p> <ul style="list-style-type: none"> _c -> für RGB-Wert (color) _w -> für Schriftgewicht (width) _s -> für Schriftgröße (size) _d -> für Text-Dekoration (decoration) _i -> für Text-Stil (style)
SHOP_LAYOUT.php	savefont
<p>Beschreibung</p> <p>DA</p>	<p>Speichert die Fonteneinstellungen eines Font-Tags in der DB. Das Tag wird um folgende Erweiterungen ergänzt in der CSS-Tabelle gespeichert:</p> <ul style="list-style-type: none"> _c -> für RGB-Wert (color) _w -> für Schriftgewicht (width) _s -> für Schriftgröße (size) _d -> für Text-Dekoration (decoration) <p>Argumente: Font-tags: \$font_tag, \$c_wert, \$w_wert, \$s_wert, \$d_wert, \$i_wert</p> <p>Rückgabewert: true bei Erfolg</p>
SHOP_LAYOUT.php	rgbshow
<p>Beschreibung</p> <p>DA</p>	<p>Holt einen RGB-CSS-Wert aus der DB und gibt ihn als Teil eines HTML-Formulars mit Eingabefeldern für R,G,B aus.</p> <p>Argumente: rgb-identifizier</p> <p>Rückgabewert: HTML-Formularteil mit Eingabemöglichkeit für R,G,B</p>
SHOP_LAYOUT.php	getnocomma
<p>Beschreibung</p> <p>DA</p>	<p>Holt einen CSS-Wert aus der DB und entfernt allfällig vorhandene Kommas. Wird für das Fontset benötigt.</p> <p>Argumente: CSS-Identifizier</p> <p>Rückgabewert: CSS-String ohne Kommas</p>

Modul: PHP-Funktionen – Shop Layout	
SHOP_LAYOUT.php	rgbdechex
Beschreibung DA	<p>Wandelt die dezimal in einem Array übergebenen Farbkomponenten (R,G,B) in den für das CSS-File notwendigen Hex-String um.</p> <p>Argumente: Farbkomponenten-Array</p> <p>[0] = R {0..255}</p> <p>[1] = G {0..255}</p> <p>[2] = B {0..255}</p> <p>Rückgabewert: RGB-Hex-String im Format (#RRGGBB)</p>
SHOP_LAYOUT.php	rgbhexdec
Beschreibung DA	<p>Wandelt einen Hex-String in der Form #RRGGBB in die entsprechenden Farbkomponenten (R,G,B) im Dezimalformat um und gibt diese in einem Array zurück:</p> <p>Argumente: HTML-Hex-String Format: # RRGGBB</p> <p>Rückgabewert: Array mit den Farbkomponenten R,G,B im Dezimalformat</p> <p>[0] = R</p> <p>[1] = G</p> <p>[2] = B</p>

Modul: Aufruf-Module – Artikel Handling	
Datei	Variable \$darstellen =
USER_ARTIKEL_HANDLING_AUFRUF.php	1
Beschreibung DA	Alle Artikel einer Kategorie im Content-Frame darstellen. Wird das Artikel-Thumbnail angeklickt, geht ein Pop-Up Fenster auf, worin das Artikelbild in Originalgrösse angezeigt wird.
USER_ARTIKEL_HANDLING_AUFRUF.php	30
Beschreibung DA	Artikel Suchen 1/2. Eingabefenster einblenden (Eingabe von einem oder mehreren Suchstrings, Bilder anzeigen Ja/Nein), danach weiter im darstellen = 3.
USER_ARTIKEL_HANDLING_AUFRUF.php	3
Beschreibung DA	Artikel Suchen 2/2: Verarbeitung der Eingabe und Ausgabe des formatierten Resultats
USER_ARTIKEL_HANDLING_AUFRUF.php	4
Beschreibung DA	Im left-Frame alle Kategorien zur Auswahl anzeigen (Auswahlmnü). Die selektierte Kategorie oder Unterkategorie wird in der Variable \$selected übergeben. Darin steht die Kategorie_ID der entsprechenden Kategorie, resp. Unterkategorie. (Diese Funktion wird von ../index.php aufgerufen -> siehe dortiges Frameset)

Modul: Aufruf-Module – Userseitiges Bestellwesens Teil 1/2	
Datei	Variable \$darstellen =
USER_BESTELLUNG_AUFRUF.php	1
Beschreibung DA	Anzeige des Warenkorbs des Kunden im Content-Frame.
USER_BESTELLUNG_AUFRUF.php	2
Beschreibung	Wird aufgerufen, sobald der Kunde einen Artikel in seinen Warenkorb gelegt hat. (Weiche: Zurück zu Artikel, in Warenkorb- oder zur Kasse gehen)
USER_BESTELLUNG_AUFRUF.php	3
Beschreibung	Wenn ein Kunde in seinem Warenkorb bei einem Artikel auf löschen klickt, wird dieser Meldungsbildschirm angezeigt

Modul: Aufruf-Module – Userseitiges Bestellwesens Teil 2/2	
Datei	Variable \$darstellen =
USER_BESTELLUNG_1.php	1

Modul: Aufruf-Module – Userseitiges Bestellweszen Teil 2/2	
Beschreibung DA	Anzeige des Warenkorbs des Kunden im Content-Frame. Bevor der Login-Screen ausgegeben wird, prüfen ob sich der Kunde schon eingeloggt hat: ja --> direkt darstellen = 11 aufrufen nein --> Login-Screen ausgeben (darstellen=1) Login-Screen für Kunde, falls nicht zu darstellen = 11 gesprungen wurde.
USER_BESTELLUNG_1.php	9
Beschreibung DA	Eingabeformular, falls jemand sein Passwort vergessen hat.
USER_BESTELLUNG_1.php	10
Beschreibung DA	Passwort einem User zuschicken
USER_BESTELLUNG_1.php	11
Beschreibung DA	Eingabeformular für Lieferadresse und Zahlungsart
USER_BESTELLUNG_1.php	2
Beschreibung DA	<ul style="list-style-type: none"> - Update der Kunden- und Bestellungsdaten. - Ausgabe der Bestellung zur Kontrolle für den Kunden (an den Browser). - Weiche für Zahlungsarten. Je nachdem ob man entweder per Vorauskasse, Rechnung, Nachnahme oder per Kreditkarte bezahlt und weiter noch, je nachdem ob die Kreditkartenzahlung intern oder extern abgewickelt wird.
USER_BESTELLUNG_1.php	3
Beschreibung DA	AGB Weiche bei Vorauskasse-, Rechnung- und Nachnahme-Zahlungen: Allgemeine Geschäftsbedingungen aus Datenbank auslesen und Darstellen. Die Bestellung wird erst abgesendet, wenn der Shopbenutzer die AGB's akzeptiert hat (danach geht es bei darstellen = 4 weiter).
USER_BESTELLUNG_1.php	5
Beschreibung DA	Kreditkarten Handling <i>INTERN</i> : Kreditkartendaten Eingabemaske und Weiterleitung an darstellen = 7 für AGB Abfrage.
USER_BESTELLUNG_1.php	6
Beschreibung DA	Kreditkartenhandling <i>EXTERN</i> : Hier wird die Funktion payment_extern (\$Pay) aufgerufen. Ihr wird im Argument eine Pay mitgegeben (siehe auch pay_def.php). Darin befinden sich alle nötigen Informationen, um eine Zahlung durchzuführen. Weiter geht es bei darstellen = 4.
USER_BESTELLUNG_1.php	7

Modul: Aufruf-Module – Userseitiges Bestellwesens Teil 2/2	
Beschreibung DA	Wenn wir eine interne Kreditkartenzahlung haben, so wird uns die Variable GNUPG mit dem Wert 1 von darstellen = 5 her übergeben. Wir überprüfen also zuerst die Kreditkarten Nummer. Falls sie falsch ist oder das Ablaufdatum ungültig ist, senden wir sie zurück und der Kunde soll sie nochmals eingeben. Zur Erkennung geben wir das Flag \$Kreditkarten_Nummer_ungueltig = 1 mit. Wenn alles gut ging, geht es weiter bei darstellen = 4.
USER_BESTELLUNG_1.php	4
Beschreibung DA	<ul style="list-style-type: none"> - Unterscheidung des Workflows je nach Zahlungsart: - Vorkasse, Rechnung, Nachnahme: AGB akzeptiert?, weiter - Interne Kreditkartenzahlung: Nummer validieren, GNU-PG enable, weiter - Externe Kreditkartenzahlung: E-Mail String zurücklesen, weiter - Bestellung an Kunden und Shopinhaber mailen - Danke an Shopuser ausgeben (mit Kontaktierungsmöglichkeit bei Fragen) - Bestellung abschliessen - Alle alten Session-ID's aus Datenbank löschen

Modul: Aufruf-Module – pop_up.php / pop_up_admin.php	
Datei	Zweck
pop_up.php / pop_up_admin.php	
Beschreibung DA	Ein grosses Bild eines Artikels in einem Fenster darstellen. Mit der Möglichkeit, das Fenster per Link zu schliessen. Diese Funktion musste als eigene Datei ausprogrammiert werden (wegen HTML).

Modul: Aufruf-Module - Administrationsaufrufe	
Datei	Variable \$darstellen =
SHOP_ADMINISTRATION_AUFRUF.php	1
Beschreibung DA	Ein neuer, leerer Artikel wird erzeugt und an darstellenArtikel weitergegeben. Auf diese Weise kann man neue Artikel in die Datenbank aufnehmen.
SHOP_ADMINISTRATION_AUFRUF.php	101
Beschreibung DA	Einem Artikel eine oder mehrere Kategorien zuordnen. Dies ist die Station nach der Eingabe der Artikeldaten. Man kommt von der Darstellen-Funktion darstellenArtikel(\$Artikel) hierhin. Von hier aus geht es weiter ins Modul bild_up.php, wo die Bilder zum Artikel gespeichert werden können.
SHOP_ADMINISTRATION_AUFRUF.php	2

Modul: Aufruf-Module - Administrationsaufrufe	
Beschreibung DA	Auswahlmenü (Kategorienwahl und Artikelwahl), wenn man einen Artikel zum Bearbeiten / Löschen auswählen will. Wenn man auf eine Kategorie / Unterkategorie klickt, baut sich das Fenster neu auf und man kann die sich darin befindenden Artikel sehen. Wenn man sich hier für einen entschieden hat, geht es weiter zu darstellen = 5.
SHOP_ADMINISTRATION_AUFRUF.php	5
Beschreibung	Nachdem der Artikel bearbeitet wurde, wird hier entweder ein Update des Artikels oder das Löschen des Artikels eingeleitet. Beim Löschen wird die Funktion delArtikel (\$Artikel_ID) aufgerufen, beim Bearbeiten eines Artikels wird updArtikel (\$Artikel_ID) benutzt.
SHOP_ADMINISTRATION_AUFRUF.php	6
Beschreibung DA	Eine neue Kategorie erstellen 1/2: Eingabe von Namen und der Positions-Nr der neuen (Haupt-)Kategorie. Beschreibung und Bilddaten blieben bei dieser Shop-Version noch aussen vor.
SHOP_ADMINISTRATION_AUFRUF.php	7
Beschreibung DA	Eine neue Kategorie erstellen 2/2: Speichern der neuen (Haupt-) Kategorie.
SHOP_ADMINISTRATION_AUFRUF.php	61
Beschreibung DA	Eine neue Unterkategorie erstellen Schritt 1/2: Eingabe der Unterkategoriedaten.
SHOP_ADMINISTRATION_AUFRUF.php	71
Beschreibung DA	Eine neue Unterkategorie erstellen 2/2: Speichern der neuen Unterkategorie.
SHOP_ADMINISTRATION_AUFRUF.php	20
Beschreibung DA	Eine Kategorie auf eine andere Position verschieben 1/2: Auswahl der neuen Position
SHOP_ADMINISTRATION_AUFRUF.php	21
Beschreibung DA	Eine Kategorie auf eine andere Position verschieben 2/2: Speichern der neuen Position.
SHOP_ADMINISTRATION_AUFRUF.php	25
Beschreibung DA	Eine Unterkategorie an eine andere Position innerhalb der gleichen Kategorie verschieben 1/2: Auswahl der neuen Position.
SHOP_ADMINISTRATION_AUFRUF.php	26
Beschreibung DA	Eine Unterkategorie auf eine andere Position innerhalb der gleichen Kategorie verschieben 2/2: Speichern der neuen Position.

Modul: Aufruf-Module - Administrationsaufrufe	
SHOP_ADMINISTRATION_AUFRUF.php	30
Beschreibung DA	Eine Unterkategorie auf eine beliebige Stelle verschieben 1/2: Auswahl der neuen Position.
SHOP_ADMINISTRATION_AUFRUF.php	31
Beschreibung DA	Eine Unterkategorie auf eine beliebige Stelle verschieben 2/2: Speichern der neuen Position.
SHOP_ADMINISTRATION_AUFRUF.php	35
Beschreibung DA	Eine Kategorie umbenennen 1/2: Eingabe des neuen Kategorienamens.
SHOP_ADMINISTRATION_AUFRUF.php	36
Beschreibung DA	Eine Kategorie umbenennen 2/2: Speichern der neuen Position.
SHOP_ADMINISTRATION_AUFRUF.php	40
Beschreibung DA	Eine Unterkategorie umbenennen 1/2: Eingabe des neuen Namens.
SHOP_ADMINISTRATION_AUFRUF.php	41
Beschreibung DA	Eine Unterkategorie umbenennen 2/2: Speichern der neuen Position
SHOP_ADMINISTRATION_AUFRUF.php	45
Beschreibung DA	Eine Kategorie/Unterkategorie löschen 1/2: Bestätigungsanfrage!
SHOP_ADMINISTRATION_AUFRUF.php	46
Beschreibung DA	Eine Kategorie/Unterkategorie löschen 2/2: Löschvorgang ausführen!

Modul: Aufruf-Module – Bild in die Datenbank schreiben	
Datei	Variablenwerte
bild_up.php	Variable \$Speichern = true

Modul: Aufruf-Module – Bild in die Datenbank schreiben

Beschreibung DA	Das bedeutet, der Shop-Administrator will ein Bild zu diesem Artikel in der Datenbank speichern. Nun wird ein neues Thumbnail vom eingelesenen Bild angefertigt (aus lizenzrechtlichen Gründen kein GIF-Support) und alle Bilddaten (insgesamt 4 Attribute der Tabelle artikel) in der Datenbank gespeichert. Diese Funktionalität verlangt seit v.1.05 <i>nicht</i> mehr nach root-Rechten.
bild_up.php	Variable \$loeschen_Bild = 1, \$Speichern = false
Beschreibung	Der Shop-Administrator hat sich in diesem Fall entschieden, das vorhandene Bild zu löschen. Dies wird nun ausgeführt.

Modul: Aufruf-Module – Bestellungsmanagement

Datei	\$darstellen =
SHOP_BESTELLUNG.php	10
Beschreibung DA	Kundendaten und seine Bestellung der Referenznummer nach anzeigen. (Hier kann man Änderungen am Kundenprofil vornehmen / die Bestellung löschen).
SHOP_BESTELLUNG.php	11
Beschreibung DA	Einen zuvor gewählten Artikel aus der aktuell bearbeiteten Bestellung löschen.
SHOP_BESTELLUNG.php	12
Beschreibung DA	Die aktuelle Bestellung löschen oder sie updaten (abspeichern). Wenn die Variable \$Speichern gleich "Speichern" ist, soll upgedated werden, sonst wird gelöscht.
SHOP_BESTELLUNG.php	13
Beschreibung DA	Hier kann man alle abgeschlossenen, aber noch nicht gespeicherten Bestellungen in einer Tabelle ansehen und per Klick darauf eine Bestellung editieren.
SHOP_BESTELLUNG.php	<i>ungleich 10, 11, 12, 13</i>
Beschreibung DA	Suchformular anzeigen. Momentan kann man hier nur nach Referenznummern suchen.

Modul: Aufruf-Module – Kundenattribute Management

Datei	\$darstellen =
SHOP_KUNDE.php	10
Beschreibung DA	Abspeichern der Einstellungen.
SHOP_KUNDE.php	<i>ungleich 10</i>
Beschreibung DA	Formular für das Mutieren der Kundenattribute (Datenfelder eines Kunden-Datensatzes) anzeigen. Hier können auch die selbstdefinierten Zusatzfelder konfiguriert werden.

Modul: Aufruf-Module – Layout Management

Datei	\$darstellen =
-------	----------------

Modul: Aufruf-Module – Layout Management	
SHOP_LAYOUT.php	10
Beschreibung DA	Speichert die veränderten Layout-Parameter in die Datenbank und ruft die Funktionen zur Erzeugung der CSS-Dateien, sowie von index.php (beinhaltet Frameset) auf.
SHOP_LAYOUT.php	20
Beschreibung DA	Gibt ein Formular aus, in welchem man per Browse-Button eine Bild-Datei auf der Festplatte auswählen kann. Diese wird dann als eines der folgenden Bilder hochgeladen: Shoplogo, linkes-, oberes-, content-Hintergrundbild.
SHOP_LAYOUT.php	21
Beschreibung DA	Dies ist die Folgeseite von darstellen = 20. Hier wird der Dateiname der hochgeladenen Datei analysiert. Handelt es sich um ein File vom Typ .gif, .jpg, .jpeg oder .png so wird sie in die entsprechende Bilddatei im Verzeichnis "Bilder" hineinkopiert. Auch für diese Bild-Funktion werden <i>keine</i> root-Rechte mehr benötigt.
SHOP_LAYOUT.php	30
Beschreibung DA	Gibt ein Formular aus, in welchem man per Browse-Button eine Bild-Datei auf der Festplatte auswählen kann, mit welcher man einen Shopbutton ersetzen kann.
SHOP_LAYOUT.php	31
Beschreibung DA	Dies ist die Folgeseite von darstellen = 30. Analysiert wird der Dateiname des hochgeladenen Files. Handelt es sich um eine Datei vom Typ .gif, so wird der gewählte Button ersetzt (gespeichert). Die Analyse erfolgt in JavaScript.
SHOP_LAYOUT.php	<i>ungleich 10, 20, 21, 30, 31</i>
Beschreibung DA	Gibt die Eingabemaske für die Layout Settings als Formular aus (mit den momentan eingestellten Werten aus der Datenbank eingefüllt).

Modul: Aufruf-Module – Allgemeine Shopeinstellungen	
Datei	\$darstellen =
SHOP_SETTINGS.php	10
Beschreibung DA	Weiche um zu unterscheiden, ob das Formular abgeschickt wurde (Speichern-Button). Shop-Settings aufbereiten und abspeichern.
SHOP_SETTINGS.php	<i>ungleich 10</i>
Beschreibung DA	Formular zum Editieren der Shop-Settings.

Modul: Aufruf-Module – Versandkosten Management	
Datei	\$darstellen =
SHOP_VERSANDKOSTEN.php	10
Beschreibung DA	Hier werden die Versandkostensettings gespeichert (Funktion: setversandkostensettings(..)) und es wird eine Erfolgsmeldung ausgegeben.

Modul: Aufruf-Module – Versandkosten Management

SHOP_VERSANDKOSTEN.php	<i>ungleich 10</i>
Beschreibung DA	Formular in welchem man die Versandkostensettings editieren kann.

Modul: Darstellungs-Module - Bestellung

Datei	Funktionsname
USER_BESTELLUNG_DARSTELLUNG.php	darstellenBestellung
Beschreibung DA	<p>Diese Funktion stellt die im Argument gelieferte Bestellung in HTML dar und leitet die Daten dann zur Übertragung in die Datenbank weiter. Ist der übergebene löschen-Wert true, dann wird die Möglichkeit angezeigt, Artikel aus dem Warenkorb zu entfernen (ist bei der Darstellung des Warenkorbes zusammen mit den Adressdaten der Bestellung nicht mehr erwünscht!). Ist das Flag Admin = true, so werden für die Buttons andere Pfade benutzt.</p> <p>Damit die Bestellung für immer statisch bleibt und die Preise nicht jedes Mal neu berechnet werden, wenn man im Bestellungsmanagement die Bestellungen editiert, musste noch ein weiteres Flag eingebaut werden, welches die Funktion berechneversandkosten(.) ausklammert.</p> <p>Argumente: Bestellungen-Objekt, löschen-Flag, Administrator-Flag</p> <p>Rückgabewert: Darstellung des Warenkorbes in HTML</p>
USER_BESTELLUNG_DARSTELLUNG.php	darstellenStringBestellung
Beschreibung DA	<p>Diese Funktion schreibt den Warenkorb-Inhalt formatiert in einen String, der zum Versand der E-Mails an den Shopbenutzer und -betreiber gebraucht wird. Funktionalität ansonsten ähnlich wie Funktion 'darstellenBestellung'.</p> <p>Argumente: Bestellungen-Objekt</p> <p>Rückgabewert: Abgefüllte Bestellung in einem <i>String</i></p>

Modul: Darstellungs-Module - Artikel

Datei	Funktionsname
SHOP_ADMINISTRATION_ARTIKEL.php	darstellenArtikel
Beschreibung DA	<p>Diese Funktion stellt den im Argument gelieferten Artikel dar und leitet die Daten dann zur Übertragung in die Datenbank weiter an die Kategorien- und die Bild-Eintragung.</p> <p>Am Ende des Eingabeformulars werden noch Buttons (Reset, Submit, Abbrechen, Hilfe) zur Navigation dargestellt.</p> <p>Argumente: Ein Artikel als Artikel-Objekt (Definition siehe artikel_def.php)</p> <p>Rückgabewert: HTML-Teil einer Page (Formular, ohne Head und ohne Body-Tags)</p>

Modul: Darstellen Funktionen – Bilder aus der Datenbank auslesen

Datei	Funktionsname
bild_view.php	bild_view
Beschreibung DA	<p>Diese Datei wird nur innerhalb des HTML-Tags verwendet. Es liest aus der Datenbank das entsprechende Bild aus und stellt es dann bereit.</p> <p>Die Datei ruft ihre einzige Funktion selbst auf.</p> <p>Argumente: Artikel_ID (INT), Bildformat (String), Bildgrösse (enum('gross', 'klein'))</p> <p>Rückgabewert: Bild</p>


Modul: Darstellen Funktionen – Hilfe aus der Datenbank auslesen

Datei	Funktionsname
USER_ADMIN_HILFE.php, ADMIN_HILFE.php	get_Hilfe
Beschreibung	<p>Diese Funktion liefert einen String (inkl. HTML-Formatierung), der den entsprechenden Hilfetext der angeforderten Seite beinhaltet.</p> <p>Die zu übergebende Hilfe_ID besteht aus dem Namen der anfordernden Datei und hat angehängt einen Suffix von _1, _2...</p> <p>Argumente: Hilfe_ID</p> <p>Rückgabewert: Hilfetext zur aktuellen Seite (HTML-String)</p>

Modul: Backup Backup Menü / Backup / Restore

Datei	Funktionalität
SHOP_BACKUP.php, SHOP_BACKUP_f1.php, ADMIN_backup.php, ADMIN_restore.php [, index.php]	<p>Backup und Restore Untermenü</p> <p>[automatisiertes] Datenbank Backup</p> <p>Restore / Download eines Datenbank Backups</p>
Beschreibung DA	<p>Die Datenbank Backuplösung des PhPepperShops verteilt sich auf vier Dateien. SHOP_BACKUP und SHOP_BACKUP_f1.php stellen dem Benutzer ein Backup Untermenü und eine Backup Konfigurationsmaske zur Verfügung.</p> <p>Die eigentliche Backup-Funktionalität und die Restore-Maske befinden sich in den Dateien ADMIN_backup.php und ADMIN_restore.php.</p> <p>Wurde das Backup als periodisch konfiguriert, wird die Datei index.php angepasst (Der String liegt in der Datenbank in der Tabelle css_file unter der ID backup).</p>

Modul: Darstellen Funktionen – Payment Schnittstelle	
Datei	Funktionsname
payment_interface.php	payment_extern
<p>Beschreibung</p> <p>DA</p>	<p>Diese Funktion liefert nicht im üblichen Sinne (mittels Return-Wert) einen Rückgabewert zurück, sondern gibt direkt mittels des echo Kommandos auf das Webdokument ein Formular aus. Dieses Formular beinhaltet, wie oben erwähnt, hidden-Fields, welche die Daten zur Zahlung beinhalten. Will man weitere Daten an das externe Zahlungssystem senden, kann man diese aus dem übergebenen Pay-Objekt herausholen und in einem weiteren hidden-Field dem externen Zahlungssystem mit übergeben. Um zu verstehen, was alles in einem Pay-Objekt vorhanden ist, siehe folgende Klassendefinitionen oder die Beschreibung im Kapitel 2.8.2):</p> <p>--> pay_def.php --> kunde_def.php --> bestellung_def.php --> artikel_def.php --> kategorie_def.php</p> <p>Diese Funktion besteht im wesentlichen aus vier Teilen:</p> <ol style="list-style-type: none"> 1.) Definition des URLs des externen Zahlungssystems. Auf diese Adresse wird gesprungen, wenn der Kunde die allgemeinen Geschäftsbedingungen akzeptiert. 2.) Auspacken der benötigten Variablen aus dem übergebenen Pay-Objekt 3.) Abfüllen der zu übergebenen Werte in ihre hidden-Fields 4.) Rückgabewert auf true setzen <p>Argumente: Pay-Objekt</p> <p>Rückgabewert: true bei Erfolg, sonst Abbruch mit der PHP-Funktion 'die'</p>

Modul: Administrations Menü	
Datei	Aufruf
Shop_Einstellungen_Menu_1.php	
 <p>DA</p>	SHOP_ADMINISTRATION_AUFRUF.php?darstellen=1
	SHOP_ADMINISTRATION_AUFRUF.php?darstellen=2&up_loe=1
	SHOP_ADMINISTRATION_AUFRUF.php?darstellen=2&up_loe=0
	Shop_Einstellungen_Menu_Kategorien.php
	SHOP_BESTELLUNG.php (nur vorhanden, falls eingeschaltet)
	SHOP_SETTINGS.php
	SHOP_LAYOUT.php
	SHOP_LAYOUT.php?darstellen=20
	SHOP_LAYOUT.php?darstellen=30
	SHOP_VERSANDKOSTEN.php
SHOP_BACKUP.php	
index.html	
USER_ADMIN_HILFE.php?Hilfe_ID=Shop_Einstellungen_Menu_1	

Modul: HTML-Darstellung	
Datei	
index.html / index.php	
Beschreibung	Das index.html leitet direkt auf index.php weiter. Dort wird dem Surfer bereits eine Session und somit eine leere Bestellung zugeordnet. Nun kann er den Shop vollumfänglich benutzen.
DA	Die Datei index.php baut noch das Frameset.
top.php / content.html	
Beschreibung	Bilden des Framesets (FRAME = 'left' wird von einer Aufruf-Funktion mit Inhalt gefüllt).
DA	Das File content.html hält einen Willkommens-Bildschirm bereit. top.php zeigt den Shop-Namen an und bietet dem Kunden Navigations-elemente wie Warenkorb, Kasse und Hilfe. Ausserdem versteckt sich dort hinter dem kleinen Stern der Passwort geschützte Eingang zur Shop-Administration.

2.3 Fehlerbehandlung in den Modulen

Allgemeine Sicherheitsvorkehrungen

Funktionen geben immer einen Rückgabewert, welcher entweder definiert ist oder aber true ist.

Wenn eine Funktion mit der Datenbank kommuniziert, wird nach jeder Verbindung überprüft, ob die Kommunikation erfolgreich durchgeführt werden konnte.

Fehlercode

Entsteht in einer Funktion ein Fehler, so bricht diese mit dem PHP-Befehl 'die("Meldung")' ab. Die Meldung ist speziell formatiert: Zuerst wird der **Modul-Code** ausgegeben. Er setzt sich aus den ersten Buchstaben der Modulnamen zusammen. Die **Meldung** enthält eine genauere Beschreibung des Fehlers, des Ortes oder des Zustandes von lokalen Variablen. Danach folgt, meist in runden Klammern geschrieben, der **Name der Funktion**, in welcher der Fehler aufgetreten ist. Optional ist der vierte Teil einer Fehlermeldung. Er wird meist nur bei komplexeren oder grösseren Funktionen verwendet. Hier werden **funktionsinterne Angaben** mitgegeben, z.B. wo in der Funktion der Fehler aufgetreten ist, welches SQL den Fehler verursacht hat oder ähnliches.

Bei Fehlern mit SQL-Queries geben die Funktionen zudem auch die Query selbst noch aus. Auch der Datenbank-Wrapper (database.php) wird in einem Fehlerfall die nicht erfolgreiche Query zur Sicherheit nochmals anzeigen.

Beispiele

Modul-Code	Meldung	Funktion	Funktionsintern
U_B_Error	Konnte die Bestellung nicht dem Kunden zuweisen	gibBestellung_an_Kunde	-
S_A_Error	(RS_ukat) Eine Unterkategorie konnte nicht entfernt werden	delKategorie	-

<i>Modul-Code</i>	<i>Meldung</i>	<i>Funktion</i>	<i>Funktionsintern</i>
S_A_Error	RS ist kein Objekt->Abbruch	katposschieben	Del_Kat_1_1

U_B_Error kommt also von USER_BESTELLUNG.php, S_A_Error kommt von SHOP_ADMINISTRATION.php. Bei USER_BESTELLUNG_DARSTELLUNG.php würde man U_B_D_Error schreiben. Wir haben aber auch an Namenskonflikte gedacht. bei SHOP_ADMINISTRATION_AUFRUF.php und SHOP_ADMINISTRATION_ARTIKEL.php wird der uneindeutige Buchstabe einfach ausgeschrieben.

3 Session Management

Das Session-Management wird über die PHP4-interne Session-Verwaltung abgewickelt und wurde grundsätzlich ohne Veränderung aus der Projektarbeit übernommen.

PHP4 ist grundsätzlich in der Lage, die Verknüpfung der Zugriffe über URL oder Cookies vorzunehmen. Die Session Konfiguration läuft weitgehend über die Datei `php.ini` ab (bei uns im Verzeichnis `/etc/`). Dort belassen wir alle Session relevanten Einstellungen auf den Default-Werten. Auf diese Weise konfiguriert, arbeitet unser Session-Management mit Cookies, deren Lebenszeit auf die Laufzeit des Browsers beschränkt ist.

3.1 Maximale Session Dauer

Weiter wird in der `php.ini` die maximale Zeit einer Session auf 1440 Sekunden, also rund 24 Minuten, beschränkt (Default Einstellung). In den Shop-Settings kann der Shop-Administrator aber die Zeit einer Session verändern. Was immer er in diesem Feld eingibt, wird nur zum Tragen kommen, solange die dort angegebene maximale Session-Zeit kürzer oder gleich lang ist, als die in der `php.ini`. Weiter ist zu beachten, dass die Zeit, die in den Shop-Settings eingestellt ist, sich nicht "direkt" auf die Session-Variable auswirkt.

Es gibt PHP-Funktionen mit welchen man die Variablen in der `php.ini` ändern könnte und somit auch die obere Limite flexibler setzen könnte. Auf den Einsatz einer solchen Funktion haben wir aber verzichtet, da kaum ein Provider Freude darüber empfindet, wenn seine Kunden in seiner `php.ini` Änderungen vornehmen könnten. Ausserdem hat man als Provider-Kunde keinen schreibenden Zugriff auf die `php.ini`.

Wenn ein Kunde den Shop betritt, erhält er nicht nur eine Session-ID, sondern auch gleich eine leere Bestellung. Dort drin gibt es ein Feld namens `expired`. Darin wird die Summe aus der aktuellen Zeit und der in den Shop-Settings definierten maximalen Session-Dauer abgespeichert. Bei jeder neuen Seite, die den Session-Management-Header (siehe weiter unten) beinhaltet, wird geprüft, ob diese Zeit nicht schon abgelaufen ist. Ist dies der Fall, so wird die Bestellung in der Datenbank gelöscht und der Kunde erhält eine neue, leere Bestellung.

Rekapitulierend kann man also sagen: Der Kunde erhält zwar keine neue Session-ID, er erhält aber eine neue Bestellung, was für ihn zum gleichen Resultat führt. Er wird vom Shop-system als 'neuer' Kunde gesehen.

3.2 Session-Management-Header

Am Ende dieses Kapitels findet man den Code, den wir Session-Management-Header nennen. Er ist überall dort in die Files integriert worden, wo der Kunde vom Shop einen HTML-Header (also eine neue Seite) zugeschickt bekommt.

Dieser Header schaut, ob die Session-ID, bei uns `$mySession_ID` genannt, schon vorhanden ist. Dann soll die Session-ID Ausgabefunktion diese Variable auslesen. Falls die Variable noch nicht existiert, soll unsere `Session_ID` `$mySession_ID` heissen, diese Variable an die aktuelle Session gebunden werden und eine Session eröffnet werden.

Als einzige Änderung am Session Management während Diplomarbeit, mussten wir die Funktionen `schliessenBestellung(.)`, `delallexpiredSessions(.)`, `delSession(.)` erweitern. Da wir jetzt auch noch eine Kunden- und eine Kunden-Referenz Tabelle haben, muss beim Löschen einer Bestellung auch ein temporärer Kunde gelöscht werden (siehe auch Kapitel 2.11, CRM).

```
// -----  
// Session_Management: (Ueberpruefen ob eine Session_ID uebergeben wurde,  
// sonst eine Neue erzeugen  
if(empty($mySession_ID)){  
    session_name("mySession_ID");  
    if (!session_register("mySession_ID")) {  
        die("<HTML><BODY><P><H1>USER_BESTELLUNG_AUFRUF.php: ACHTUNG session_regis  
            ter() = false, konnte keine Session_ID zuteilen</H1></P><BR><BR></BODY>  
            </HTML>");  
    }  
}  
else {  
    session_id($mySession_ID);  
}
```

4 Datenbank

Das Datenbankdesign ist eine sehr zentrale Komponente bei der Entwicklung eines Web-Shop-Frameworks. Dementsprechend investierten wir viel Zeit und Energie in die fortlaufende Entwicklung dieses Designs. *Security-Aspekte* zum Datenbankdesign wurden schon im Kapitel 2.1 behandelt. Wir konnten auf einem guten Fundament aus der Praktischen Arbeit aufbauen.

4.1 Idee und Konzept

Damit wir einen möglichst dynamischen Shop, also einen Shop mit vielen Freiheiten und grossem Einsatzgebiet erhalten, haben wir uns schon in der Praktischen Arbeit für ein etwas komplexeres Design entschieden.

Weiter war es eine Grundidee, dass man den Shop *komplett* in der Datenbank hat. Dies ist vom Backup her ein Vorteil (einfach Backup der Datenbank anfertigen, fertig) und vom Zugriff her, dachten wir, auch. Leider haben wir uns vor dem Design wenig mit BLOBs in Datenbanken beschäftigt. Es erwies sich später als hinderlich, wenn man BinaryLargeObjects in der gleichen Tabelle hat wie weitere Daten. Es ist uns bewusst, dass man saubererweise die Bilder der Artikel in einer gesonderten Tabelle hätte ablegen müssen. Wir trugen dieser Erkenntnis Rechnung und haben alle weiteren Bilder (Layout Management) im Dateisystem abgelegt.

Das ursprüngliche Datenbankdesign wurde um einige Tabellen und viele Attribute ergänzt. Zum Beispiel haben wir aufgrund des CRMs die Kundendaten aus der Bestellung ausgegliedert und ihnen eine eigene Tabelle (kunde) gegeben. Bei dieser Gelegenheit haben wir auch den Kundendatensatz überarbeitet und ihm weitere Attribute gegeben (Postfach,...). Wir haben versucht, alle möglichen Adressdaten zu berücksichtigen. Dank der neuen, dynamischen Kundenattribut-Verwaltung kann man jetzt auch flexibel auswählen, mit welchen Kundenattributen man arbeiten will.

4.2 Personalisierung durch das *Installationstool config.pl*

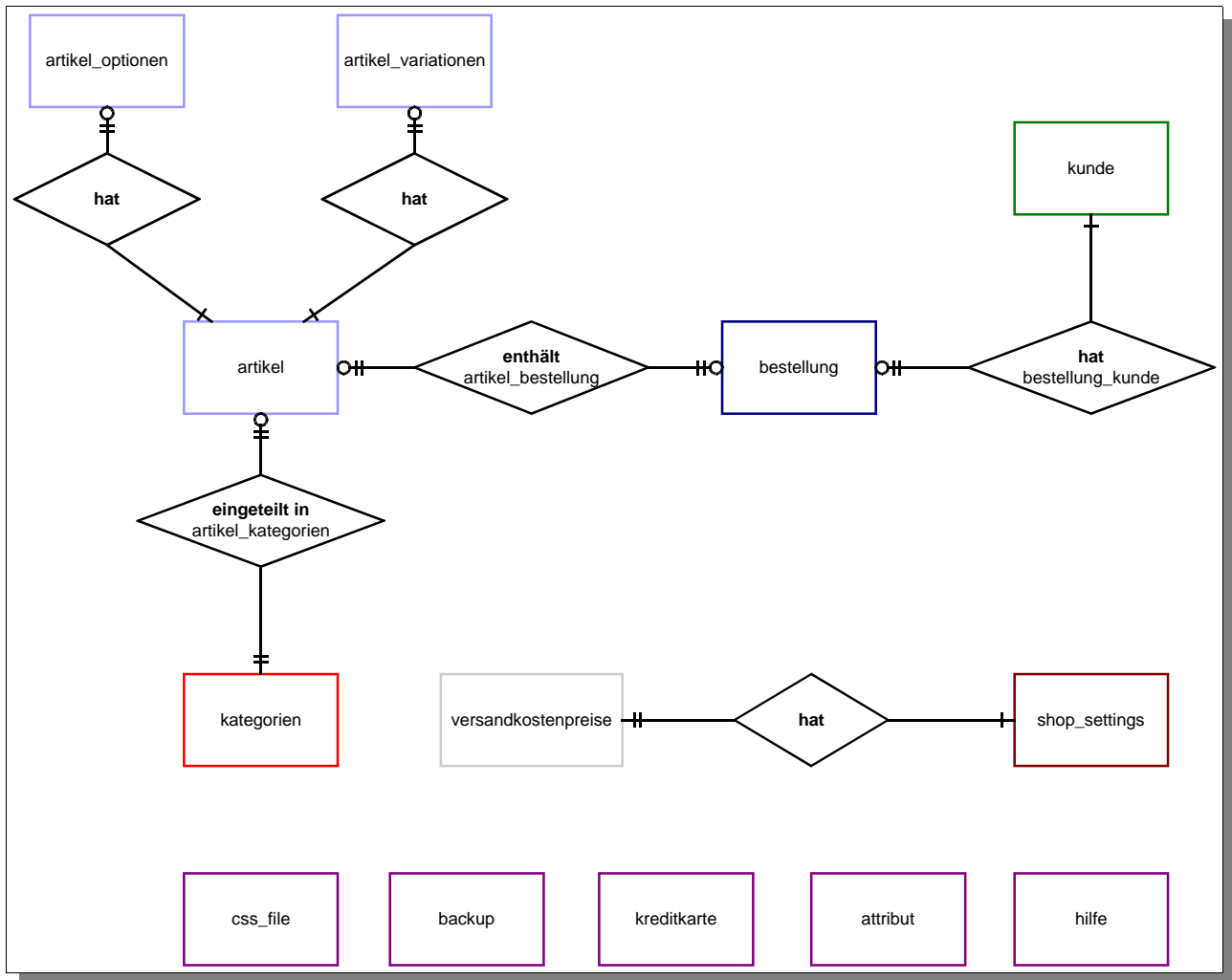
Damit mehrere Shops gleichzeitig in einem DBMS koexistieren können, müssen die Datenbanken der einzelnen Shops personalisiert sein. Der alte Shop der Praktischen Arbeit arbeitete immer mit der Datenbank namens shopdb. Mühsam musste bei jeder Installation der Shopname und der Datenbankuser angepasst werden. Dank dem Installationsscript `config.pl` wird einem diese Arbeit jetzt abgenommen.

Das Perl-Script `config.pl` erstellt folgende personalisierten Datenbank Dateien (SQL):

- `<shopname>_create.sql` Datenbank und User erstellen
- `<shopname>_inser.sql` Default-Shopdaten in die Datenbank laden
- `<shopname>_del.sql` Datenbank und User löschen

Personalisiert wird hierbei gespeichert, ob User angelegt werden sollen oder nicht und wie der Name der Datenbank lautet.

4.3 Relationen in der Datenbank (Übersicht)



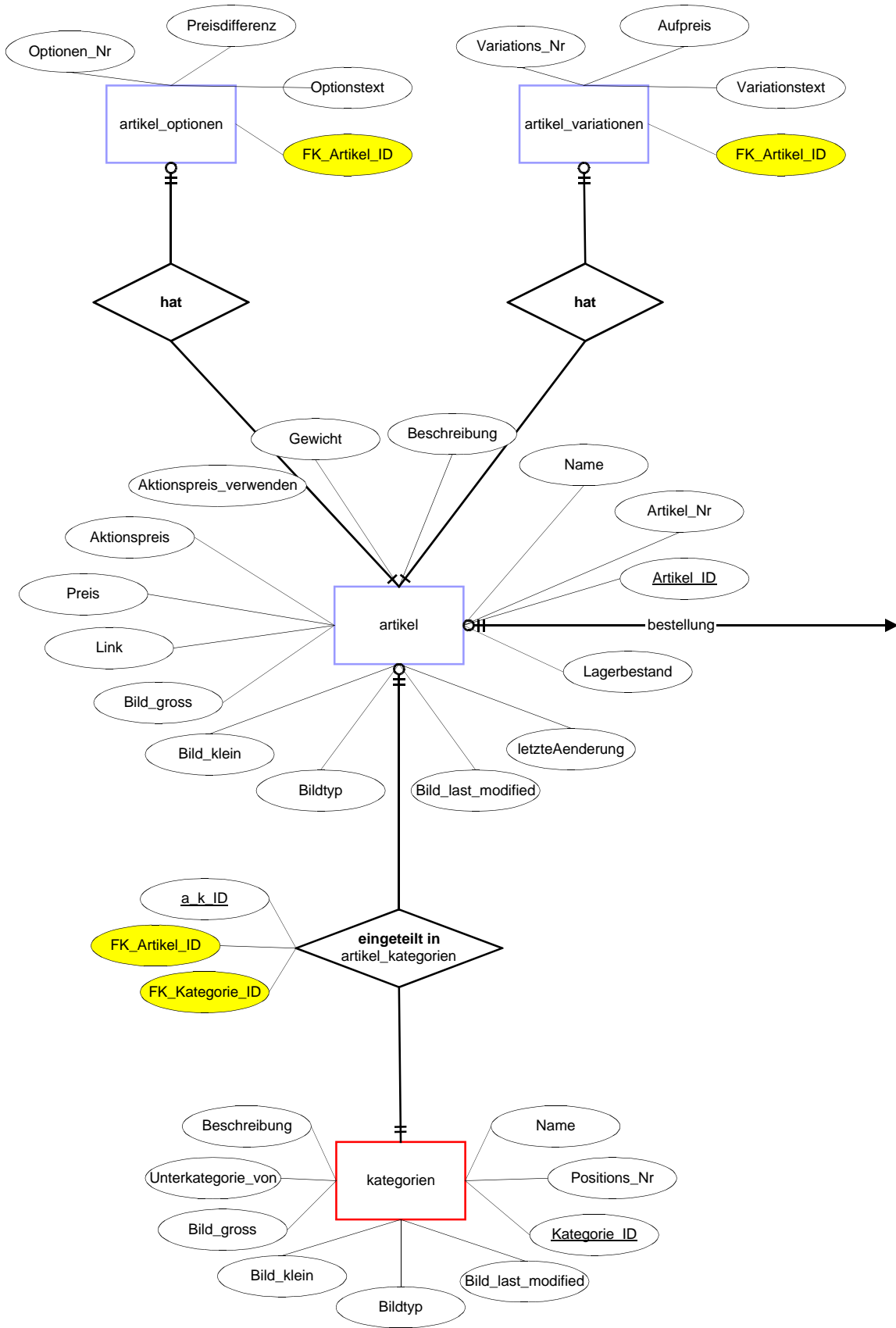
Binärer ER-Dialekt:	
○—	0..1
+—	1
⊥—	1..n
⊖⊥—	0..n

4.4 Entity Relationship Diagramm der Shop Datenbank

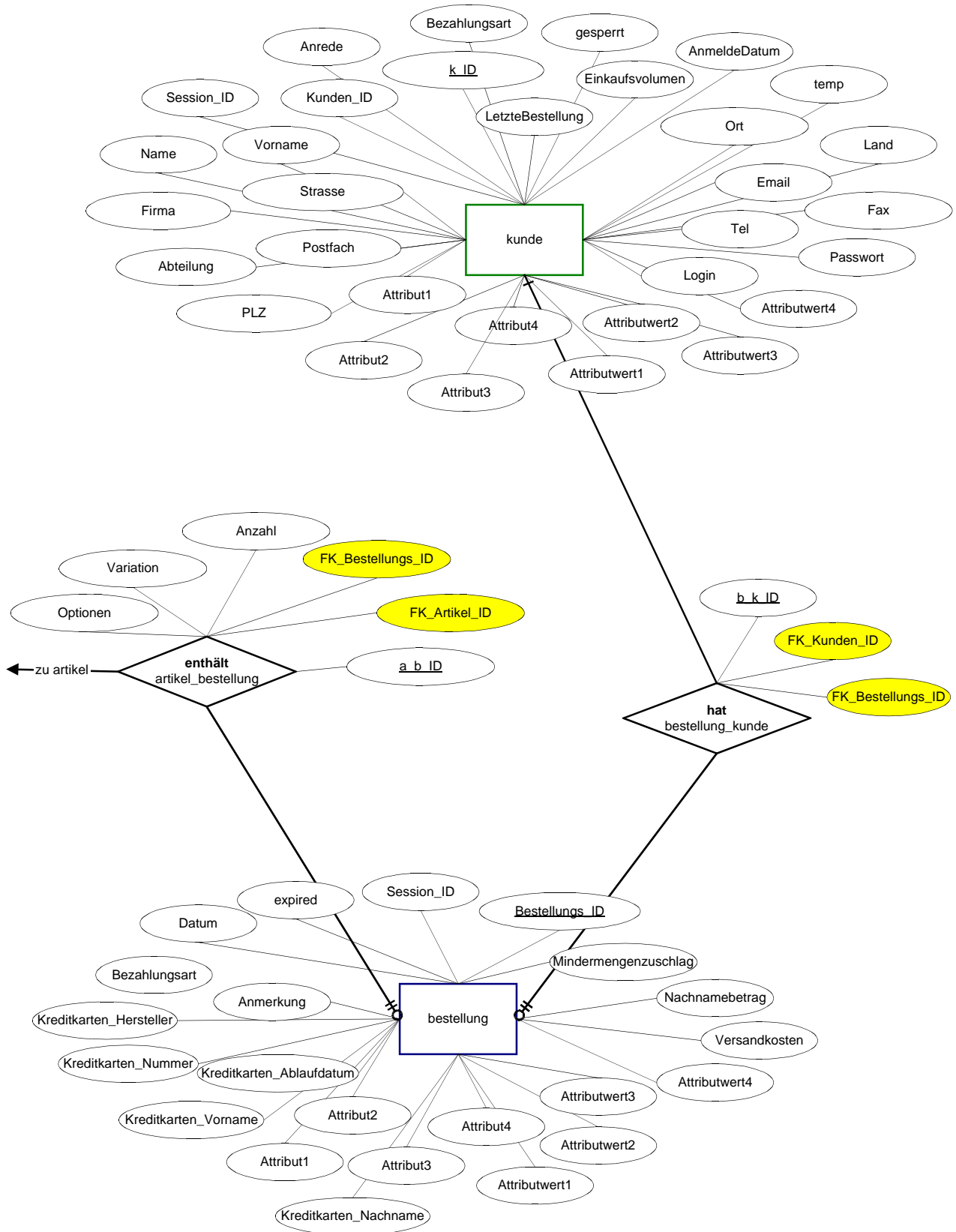
Das Entity Relationship Diagramm der PhPepperShop Datenbank wird hier inklusive allen Tabellen-Attributen dargestellt. Die Darstellung wurde so gewählt, dass die Datenbank in drei Teile aufgesplittet gezeigt wird.

Im Teil 1 sieht man den Artikel mit seinen Optionen und Variationen und der 'm zu n'-Verknüpfung zu den Kategorien. Teil 2 zeigt die Beziehung zwischen einem Kunden und seinen Bestellungen. Der dritte Teil ist losgelöst von den anderen beiden und stellt die Shop-Settings mit den Versandkostenpreisen und allen weiteren Zusatztabelle dar.

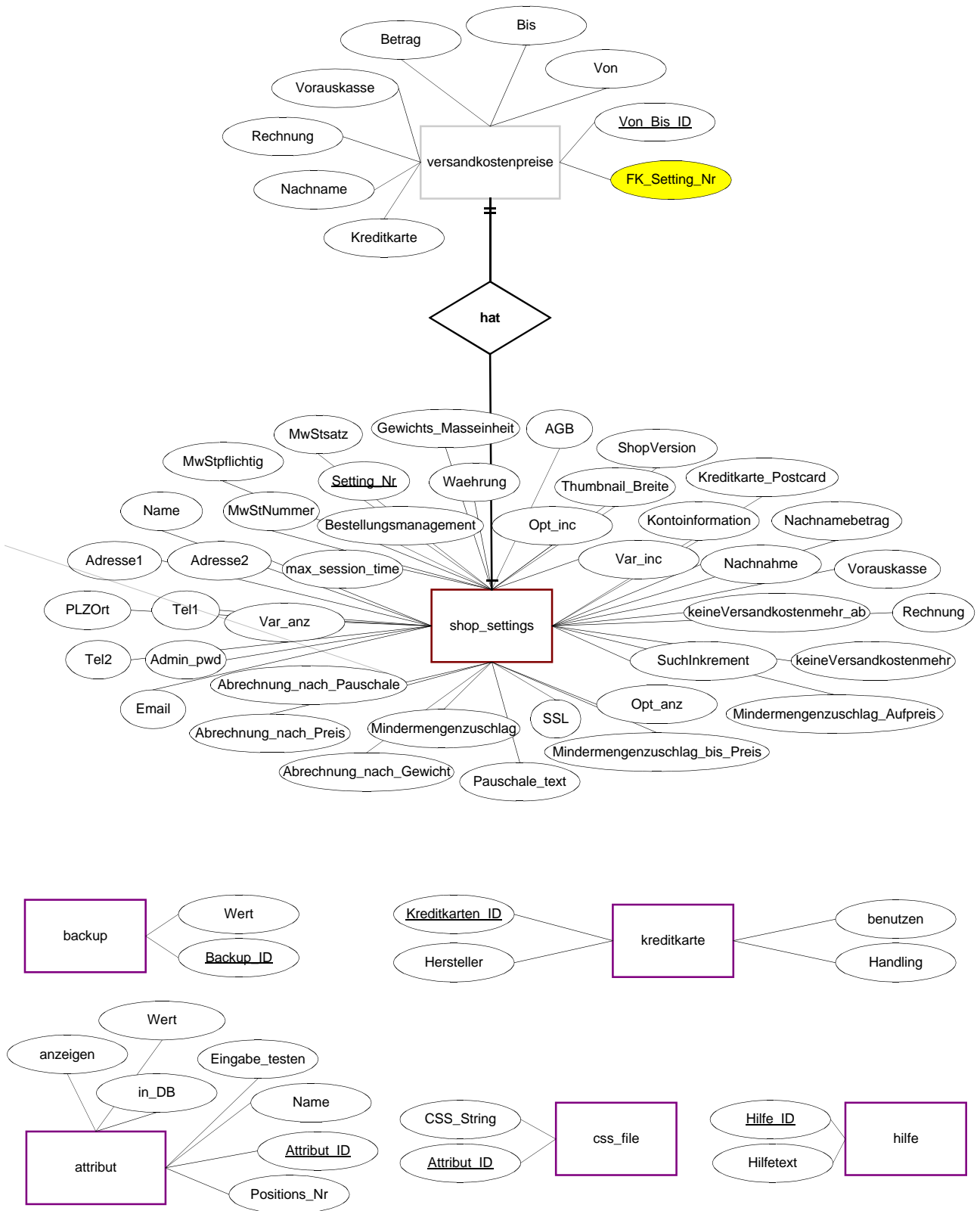
ER Diagramm Teil 1



ER Diagramm Teil 2



ER Diagramm Teil 3



4.5 Tabellen und ihre Attribute im Detail

Es folgen nun die Tabellen im Detail. Hier sieht man auch welche Datentypen für die jeweiligen Attribute verwendet wurden:

<i>artikel</i>	<i>kategorien</i>
<pre> Artikel_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, Artikel_Nr VARCHAR(128) NOT NULL, Name VARCHAR(128) NOT NULL, Beschreibung TEXT NOT NULL, Preis DOUBLE NOT NULL DEFAULT 0.0, Aktionspreis DOUBLE NOT NULL DEFAULT 0.0, Aktionspreis_verwenden enum('N','Y') NOT NULL DEFAULT 'N', Gewicht DOUBLE NOT NULL DEFAULT 0.0, Link VARCHAR(255), Bild_gross MEDIUMBLOB, <i>/* Achtung: Bei anderer DB, Typ ev. = BLOB */</i> Bild_klein BLOB, Bildtyp VARCHAR(15), Bild_last_modified TIMESTAMP(14), Lagerbestand INT NOT NULL DEFAULT 0, letzteAenderung DATE, PRIMARY KEY (Artikel_ID), UNIQUE UC_Artikel_ID (Artikel_ID) </pre>	<pre> Kategorie_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, Positions_Nr INT NOT NULL DEFAULT 0, Name VARCHAR(128) NOT NULL, Beschreibung TEXT NOT NULL, Bild_gross MEDIUMBLOB, <i>/* Achtung: Bei anderer DB, Typ ev. = BLOB */</i> Bild_klein BLOB, Bildtyp VARCHAR(15), Bild_last_modified TIMESTAMP(14), Unterkategorie_von VARCHAR(128) DEFAULT NULL, PRIMARY KEY (Kategorie_ID), UNIQUE UC_Kategorie_ID (Kategorie_ID), INDEX IDX_kategorien_1 (Name) </pre>

<i>bestellung</i>	<i>kunde</i>
<pre> Bestellungs_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, Session_ID CHAR(32) NOT NULL DEFAULT 0, <i>/* Unsere getesteten Session_IDs waren immer CHAR(32)... aber man weiss ja nie */</i> expired BIGINT NOT NULL DEFAULT 0, <i>/* In anderen DBs ev. anderer Typ (10 Stellen) */</i> Datum Date, Anmerkung TEXT, Bezahlungsart CHAR(32) NOT NULL, Kreditkarten_Hersteller CHAR(32) NOT NULL, Kreditkarten_Nummer CHAR(16) NOT NULL, Kreditkarten_Ablaufdatum VARCHAR(10) NOT NULL, Kreditkarten_Vorname VARCHAR(128) NOT NULL, Kreditkarten_Nachname VARCHAR(128) NOT NULL, Attribut1 VARCHAR(128) NOT NULL, Attribut2 VARCHAR(128) NOT NULL, Attribut3 VARCHAR(128) NOT NULL, Attribut4 VARCHAR(128) NOT NULL, Attributwert1 VARCHAR(128) NOT NULL, Attributwert2 VARCHAR(128) NOT NULL, Attributwert3 VARCHAR(128) NOT NULL, Attributwert4 VARCHAR(128) NOT NULL, Versandkosten DOUBLE NOT NULL DEFAULT 0.0, Nachnamebetrag DOUBLE NOT NULL DEFAULT 0.0, Mindermengenzuschlag DOUBLE NOT NULL, temp_message_string TEXT NOT NULL, Rechnungsbetrag DOUBLE NOT NULL DEFAULT 0.0, DEFAULT 0.0, Bestellung_abgeschlossen enum('N','Y') NOT NULL DEFAULT 'N', Bestellung_ausgeloeset enum('N','Y') NOT NULL DEFAULT 'N', Bestellung_bezahlt enum('N','Y') NOT NULL DEFAULT 'N', PRIMARY KEY (Bestellungs_ID) UNIQUE UC_Bestellungs_ID (Bestellungs_ID) </pre>	<pre> k_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, Kunden_ID VARCHAR(255) NOT NULL, Kunden_Nr VARCHAR(128) NOT NULL DEFAULT 0, Session_ID CHAR(32) NOT NULL DEFAULT 0, Anrede VARCHAR(24) NOT NULL, Vorname VARCHAR(128) NOT NULL, Nachname VARCHAR(128) NOT NULL, Firma VARCHAR(128) NOT NULL, Abteilung VARCHAR(128) NOT NULL, Strasse VARCHAR(128) NOT NULL, Postfach VARCHAR(16) NOT NULL, PLZ VARCHAR(32) NOT NULL, Ort VARCHAR(128) NOT NULL, Land VARCHAR(128) NOT NULL, Tel VARCHAR(32) NOT NULL, Fax VARCHAR(32) NOT NULL, Email VARCHAR(128) NOT NULL, Login VARCHAR(32) NOT NULL, Passwort VARCHAR(32) NOT NULL, gesperrt enum('N','Y') NOT NULL DEFAULT 'N', temp enum('N','Y') NOT NULL DEFAULT 'Y', Bezahlungsart VARCHAR(32) NOT NULL, AnmeldeDatum Date, LetzteBestellung Date, Einkaufsvolumen DOUBLE NOT NULL DEFAULT 0.0, <i>/* Attribute sind selbstkonfigurierbare Eintraege */</i> Attribut1 VARCHAR(128) NOT NULL, Attribut2 VARCHAR(128) NOT NULL, Attribut3 VARCHAR(128) NOT NULL, Attribut4 VARCHAR(128) NOT NULL, Attributwert1 VARCHAR(128) NOT NULL, Attributwert2 VARCHAR(128) NOT NULL, Attributwert3 VARCHAR(128) NOT NULL, Attributwert4 VARCHAR(128) NOT NULL, PRIMARY KEY (k_ID), UNIQUE UC_k_ID (k_ID), UNIQUE UC_Kunden_ID (Kunden_ID), UNIQUE UC_Login (Login), INDEX IDX_Kunden_ID_1 (Kunden_ID) </pre>

<i>attribut</i>	<i>artikel_kategorie</i>
<pre> Attribut_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, Name VARCHAR(128) NOT NULL, Wert VARCHAR(128) NOT NULL, anzeigen enum('N','Y') NOT NULL DEFAULT 'Y', in_DB enum('N','Y') NOT NULL DEFAULT 'Y', Eingabe_testen enum('N','Y') NOT NULL DEFAULT 'Y', Positions_Nr INT NOT NULL DEFAULT 0, PRIMARY KEY (Attribut_ID), UNIQUE UC_Attribut_ID (Attribut_ID), UNIQUE UC_Positions_Nr (Positions_Nr) </pre>	<pre> a_k_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, FK_Artikel_ID INT NOT NULL, FK_Kategorie_ID INT NOT NULL, FOREIGN KEY (FK_Artikel_ID) REFERENCES artikel (Artikel_ID), FOREIGN KEY (FK_Kategorie_ID) REFERENCES kategorien (Kategorie_ID), PRIMARY KEY (a_k_ID) </pre>

<i>artikel_optionen</i>	<i>artikel_variationen</i>
<pre> Optionen_Nr INT NOT NULL DEFAULT 1, Optionstext CHAR(64), Preisdifferenz DOUBLE NOT NULL DEFAULT 0.0, FK_Artikel_ID INT NOT NULL, FOREIGN KEY (FK_Artikel_ID) REFERENCES artikel (Artikel_ID) </pre>	<pre> Variations_Nr INT NOT NULL DEFAULT 1, Variationstext CHAR(64), Aufpreis DOUBLE NOT NULL DEFAULT 0.0, FK_Artikel_ID INT NOT NULL, FOREIGN KEY (FK_Artikel_ID) REFERENCES artikel (Artikel_ID) </pre>

<i>artikel_bestellung</i>	<i>bestellung_kunde</i>
<pre> a_b_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, FK_Artikel_ID INT NOT NULL, FK_Bestellungs_ID INT NOT NULL, Anzahl INT NOT NULL DEFAULT 1, Variation TEXT NOT NULL, /* Aufbau einer Variation: Variationstext p Aufpreis */ Optionen TEXT NOT NULL, /* Aufbau einer Variation: Optionstext p Preisdifferenz ... */ FOREIGN KEY (FK_Artikel_ID) REFERENCES artikel (Artikel_ID), FOREIGN KEY (FK_Bestellungs_ID) REFERENCES bestellung (Bestellungs_ID), PRIMARY KEY (a_b_ID,FK_Artikel_ID,FK_Bestel lungen_ID) /* Anm. Das Trennzeichen ist ALT+0254 */ </pre>	<pre> b_k_ID INT NOT NULL DEFAULT 0 AUTO_INCREMENT, FK_Kunden_ID VARCHAR(255) NOT NULL, FK_Bestellungs_ID INT NOT NULL, FOREIGN KEY (FK_Kunden_ID) REFERENCES kunde (Kunden_ID), PRIMARY KEY (b_k_ID) </pre>

<i>hilfe</i>	<i>css_file</i>
<pre> Hilfe_ID VARCHAR(128) NOT NULL, Hilfetext TEXT, PRIMARY KEY (Hilfe_ID), UNIQUE UC_Hilfe_ID (Hilfe_ID) </pre>	<pre> Attribut_ID CHAR(32) NOT NULL, CSS_String CHAR(255), PRIMARY KEY (Attribut_ID), UNIQUE UC_Attribut_ID (Attribut_ID) </pre>

<i>versandkostenpreise</i>	<i>kreditkarte</i>
<pre> Von_Bis_ID INT DEFAULT 0 NOT NULL AUTO_INCREMENT, Von DOUBLE DEFAULT 0.0, Bis DOUBLE DEFAULT 100.0, Betrag DOUBLE DEFAULT 0.0, Vorkasse enum('N','Y') NOT NULL DEFAULT 'Y', Rechnung enum('N','Y') NOT NULL DEFAULT 'Y', Nachname enum('N','Y') NOT NULL DEFAULT 'Y', Kreditkarte enum('N','Y') NOT NULL DEFAULT 'N', FK_Setting_Nr INT NOT NULL DEFAULT 1, FOREIGN KEY (FK_Setting_Nr) REFERENCES shop_settings (Setting_Nr), PRIMARY KEY (Von_Bis_ID), UNIQUE UC_Von_Bis_ID (Von_Bis_ID) </pre>	<pre> Kreditkarten_ID INT DEFAULT 0 NOT NULL AUTO_INCREMENT, Hersteller CHAR(32) NOT NULL, Handling enum('intern','extern') NOT NULL DEFAULT 'intern', benutzen enum('N','Y') NOT NULL DEFAULT 'Y', PRIMARY KEY (Kreditkarten_ID), UNIQUE UC_Kreditkarten_ID (Kreditkarten_ID) </pre>

<i>shop_settings (Teil 1/2)</i>	<i>shop_settings (Teil 2/2)</i>
<pre> Setting_Nr INT NOT NULL DEFAULT 0 AUTO_INCREMENT, MwStsatz DOUBLE NOT NULL DEFAULT 7.6, MwStpflichtig enum('N','Y') NOT NULL DEFAULT 'Y', MwStNummer INT NOT NULL DEFAULT 0, Name CHAR(48) NOT NULL DEFAULT '{shop_db} ', Adresse1 CHAR(48) NOT NULL DEFAULT 'Adresse 1', Adresse2 CHAR(48) NOT NULL DEFAULT 'Adresse 2', PLZOrt CHAR(48) NOT NULL DEFAULT 'PLZ und Ort', Tel1 CHAR(24) NOT NULL DEFAULT 'Telefon Nummer 1', Tel2 CHAR(24) NOT NULL DEFAULT 'Telefon Nummer 2', Email VARCHAR(128) NOT NULL DEFAULT 'IhrShop@shopserver.com', Admin_pwd CHAR(16) NOT NULL DEFAULT 'machshop', Abrechnung_nach_Preis enum('N','Y') NOT NULL DEFAULT 'N', Abrechnung_nach_Gewicht enum('N','Y') NOT NULL DEFAULT 'N', Abrechnung_nach_Pauschale enum('N','Y') NOT NULL DEFAULT 'Y', Pauschale_text varchar(127) NOT NULL DEFAULT 'Versand- und Verpackungskosten', Mindermengenzuschlag enum('N','Y') NOT NULL DEFAULT 'N', Mindermengenzuschlag_bis_Preis DOUBLE NOT NULL DEFAULT 0.0, Mindermengenzuschlag_Aufpreis DOUBLE NOT NULL DEFAULT 0.0, </pre>	<pre> keineVersandkostenmehr enum('N','Y') NOT NULL DEFAULT 'N', keineVersandkostenmehr_ab DOUBLE NOT NULL DEFAULT 0.0, Rechnung enum('N','Y') NOT NULL DEFAULT 'Y', Nachnahme enum('N','Y') NOT NULL DEFAULT 'Y', Vorkasse enum('N','Y') NOT NULL DEFAULT 'Y', Kreditkarten_Postcard enum('N','Y') NOT NULL DEFAULT 'N', Nachnamebetrag DOUBLE NOT NULL DEFAULT 0.0, Kontoinformation VARCHAR(255) NOT NULL DEFAULT 'Einzahlungen bitte auf unser Postkonto PC 40-168786-4', Waehrung CHAR(4) NOT NULL DEFAULT 'CHF', ShopVersion CHAR(48) NOT NULL DEFAULT 'Oktober 2001, DB Version v.1.05', Gewichts_Masseinheit CHAR(16) NOT NULL DEFAULT 'kg', Thumbnail_Breite INT NOT NULL DEFAULT 100, AGB TEXT, /* Max. 65535 Bytes gross in MySQL */ SSL enum('N','Y') NOT NULL DEFAULT 'N', Bestellungsmanagement enum('N','Y') NOT NULL DEFAULT 'Y', SuchInkrement INT NOT NULL DEFAULT 10, max_session_time BIGINT, /* In anderen DBs ev. ein anderer Typ (10 Stellen) */ Opt_inc INT NOT NULL DEFAULT 3, Var_inc INT NOT NULL DEFAULT 3, Opt_anz INT NOT NULL DEFAULT 5, Var_anz INT NOT NULL DEFAULT 5, PRIMARY KEY (Setting_Nr), UNIQUE UC_Setting_Nr (Setting_Nr) </pre>

<i>backup</i>
<pre> CREATE TABLE backup (Backup_ID CHAR(32) NOT NULL, Wert CHAR(255), PRIMARY KEY (Backup_ID), UNIQUE UC_Backup_ID (Backup_ID)); </pre>

4.6 SQL-Query Handling

Dieses Kapitel hängt eigentlich stark mit der Security (Kapitel 2.1) zusammen. Das Konzept anstatt die Query selbst in die Funktion einzubinden, dort nur Variablen zu benutzen, entstand ursprünglich in der Praktischen Arbeit dieses Frühjahrs. Dieser Ansatz hat verschiedene Vorteile: Erstens werden so die SQL-Queries von den Funktionen entkoppelt, was entscheidend ist, sobald man gedenkt eine andere Datenbank einzusetzen (und sich somit die SQL-Syntax leicht ändert). Weiter hat man so alle Queries zentral abgelegt. Bei uns in zwei Dateien und das für den ganzen Shop. Zu guter Letzt lässt sich so unsere Security gut implementieren. Wir haben dieses Konzept auch in der Diplomarbeit strikte eingehalten.

Als Anmerkung sei hier noch erwähnt, dass sich die Konvertierung der Queries am einfachsten und schnellsten mit einem Tool erledigen lässt. Wir haben hier Erstaunliches gesehen bei der Programmkombination: DATANAMICs DeZign for databases und DATANAMICs ImportER Scripts für Windows (www.datanamic.com). Diese Tools vermögen weit mehr als das einfache Konvertieren von SQLs zwischen Datenbanken-Syntax und kosten zusammen nur US\$ 180.00.

4.7 DB-Wrapper (*database.php*)

Von unserem Dozenten und Diplomarbeitbetreuer, Herr Feisthammel, erhielten wir schon zu Beginn der Projektarbeit einen Datenbank Wrapper für Sybase. Dieser Wrapper stellt einerseits eine PHP Datenbank-Connection zur Verfügung, bietet die zum Datenbank-Handling nötigen Funktionen an (Query ausführen, Resultset auswerten, Resultzeilen zählen u.s.w.) und hat andererseits für jede Datenbank, auf die zugegriffen werden soll, verschiedene Klassen. Wie schon erwähnt, war dieser Wrapper für Sybase schon fertig programmiert und getestet. Während der Projektarbeit portierten wir ihn noch auf die MySQL Datenbank.

Wir benötigen die Artikel-ID eines neu eingefügten Artikels (Funktion `newArtikel` zum Einfügen des Artikels und seiner Optionen und Variationen). Dies lässt sich per MySQL sehr leicht und effizient erledigen. Sobald bei einem `INSERT` ein neuer Autowert generiert wird, kann dieser mit dem Kommando `mysql_insert_id()` ausgelesen werden. Die `Exec`-Funktion der `TMySQLDatabase`-Klasse in `database.php` liefert also nicht mehr nur `true` oder `false` zurück sondern:

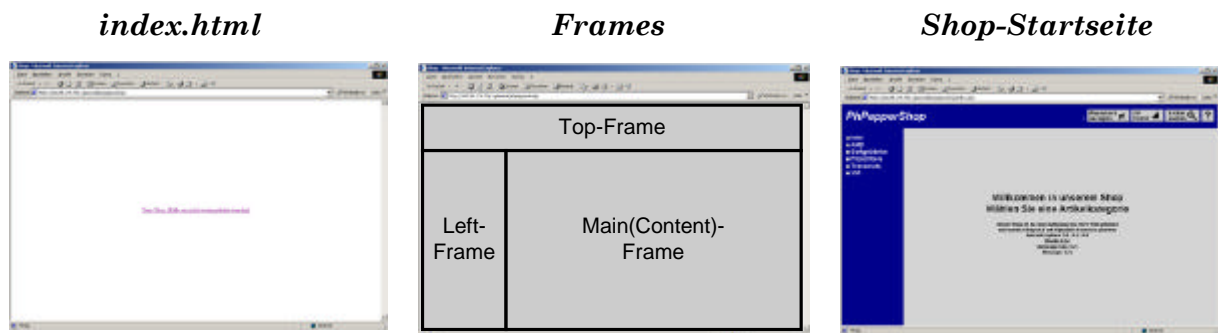
- `false`, wenn `Exec` nicht korrekt ausgeführt werden konnte
- `true`, wenn kein `Insert` mit `Auto-Increment` Funktion ausgeführt wurde
- Den zuletzt erzeugten `Auto-Increment` Wert, wenn es bei einem `Insert` so einen gegeben hat.

Diese Erweiterung wurde bei der `TSybaseDatabase`-Klasse noch nicht vorgenommen. Man muss also diesen neuen `Auto-Increment` Wert zuerst per `Select`-Anweisung anfordern und dann zurückgeben. Will man den Shop also unter Sybase laufen lassen, *muss* diese Funktion zuerst noch programmiert werden.

5 Shop Layout Management

5.1 Shop Layout Beschreibung

Zuerst wird die Datei `index.html` vom Webserver geladen und an den Browser ausgegeben. Diese Datei hat die Funktion, den Shopuser sofort per `http-redirect` auf `index.php` weiterzuleiten, welches sich im Shopverzeichnis befindet. Unterstützt der Browser diese Funktion nicht, kann sich der Surfer durch anklicken eines Links zum Shop navigieren. Ausserdem kann diese Datei noch dazu verwendet werden, Informationen für Suchrobots bereitzustellen. Wird der Shop in eine bestehende Homepage eingebettet, kann `index.html` weggelassen und der Shop direkt mit einem Link auf `index.php` aufgerufen werden.



Danach wird von `index.php` das Frameset aufgebaut und die Shopstartseite dargestellt. Der gesamte Shop (Benutzersicht) wird nun in diesem Dreierframeset dargestellt. Das oberste Frame (Top-Frame) wird verwendet, um Funktionen anzubieten, die jederzeit ausgeführt werden können (Warenkorb anzeigen, Zur Kasse, Hilfe, Suchfunktion) und um den Shopnamen oder das Shoplogo permanent anzuzeigen.

Im linken Frame (Left-Frame) werden die Artikelkategorien und Unterkategorien zur Auswahl angezeigt. Dieses Frame wird aktualisiert, sobald der Shopbenutzer eine Kategorie oder Unterkategorie anklickt.

Das grösste Frame (Content-Frame) wird zur Userinteraktion verwendet. In diesem Frame läuft praktisch die gesamte Interaktion mit dem Benutzer ab.

Beim Anklicken eines Artikelthumbnails und des Hilfe-Buttons wird ein neues Fenster geöffnet. Dieses POP-UP ist mittels JavaScript realisiert. Die entsprechenden JavaScript Funktionen befinden sich direkt im HTML-Quelltext der jeweiligen Sites. Die POP-UP-Fenster können entweder per eingblendetem Link oder mittels Schliessen-Button wieder geschlossen werden.

Es ist Shopweit ein Fontset definiert. Alle Formatierungsangaben befinden sich an zentraler Stelle in einem CSS-File, auf welches im Layout Management noch genauer eingegangen wird.

5.2 Layout Administrationsmodus

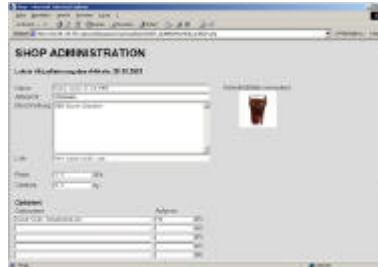
Das Benutzerinterface für den Shopadministrator überarbeiteten wir vollständig. Der Administrationsbereich besitzt nun eine eigene CSS-Datei, damit er immer gleich aussieht, auch wenn der Shopadministrator im Shoplayout-Menü Farbeinstellungen tätigt, die eine Bedienung des Shops verunmöglichen.

Da im Administrationsbereich wesentlich grössere Formulare und Menüs ausgegeben werden, entschieden wir uns, auf ein Frameset zu verzichten. Alle Menüs werden ganzseitig dargestellt, wobei in der zweitobersten Zeile immer der Menütitel angezeigt wird.

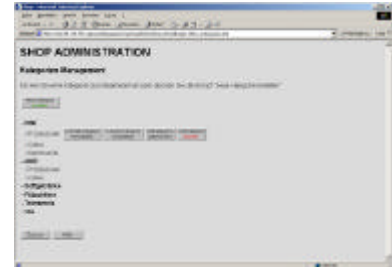
Hauptmenü



Artikel bearbeiten



Kategorienmanagement



5.3 Layout Management

Unser Ziel war es, einen Shop zu entwickeln, dessen optisches Erscheinungsbild beliebig auf die Bedürfnisse des Betreibers angepasst werden kann.

Um dies zu erreichen, müssen folgende HTML-Komponenten dynamisch verändert werden können:

- Hintergrundbilder/-Farben sämtlicher Frames
- Alle im Shop verwendeten Schriften-Tags
- Shoplogo, Shoptitel
- Shop-Buttons
- Frame-Grössen

Wir erstellten ein Layout-Management-Konzept, mit dem es möglich ist, alle Einstellungen des Layouts per Web vornehmen zu können. Dieses besteht aus folgenden drei Teilen:

Layout Management

Ruft der Shopadministrator das Layout Management Menu auf, werden in der Datenbank-tabelle `css-file` alle Shopeinstellungen ausgelesen. Diese werden in ein HTML-Formular eingefüllt, welches dann bearbeitet werden kann. Wird das Formular abgesendet, werden die Shopeinstellungen wieder in die Datenbank gespeichert. Zusätzlich werden die beiden Template-Files `indextemplate.txt` und `csstemplate.txt` eingelesen. Mit einem eigens zu diesem Zweck programmierten Parser werden die Files `index.php` und `shopstyles.css` erzeugt. Dies geschieht, indem spezielle Tags, welche sich in eckigen Doppelklammern befinden (`<<Tagname>>`), durch entsprechende Shopeinstellung aus der Datenbank ersetzt werden:

Zeile im Template-File:

```
body.left { color:<<left_font_c>>; background:<<left_bg_c>>; <<left_bg_img>> }
```

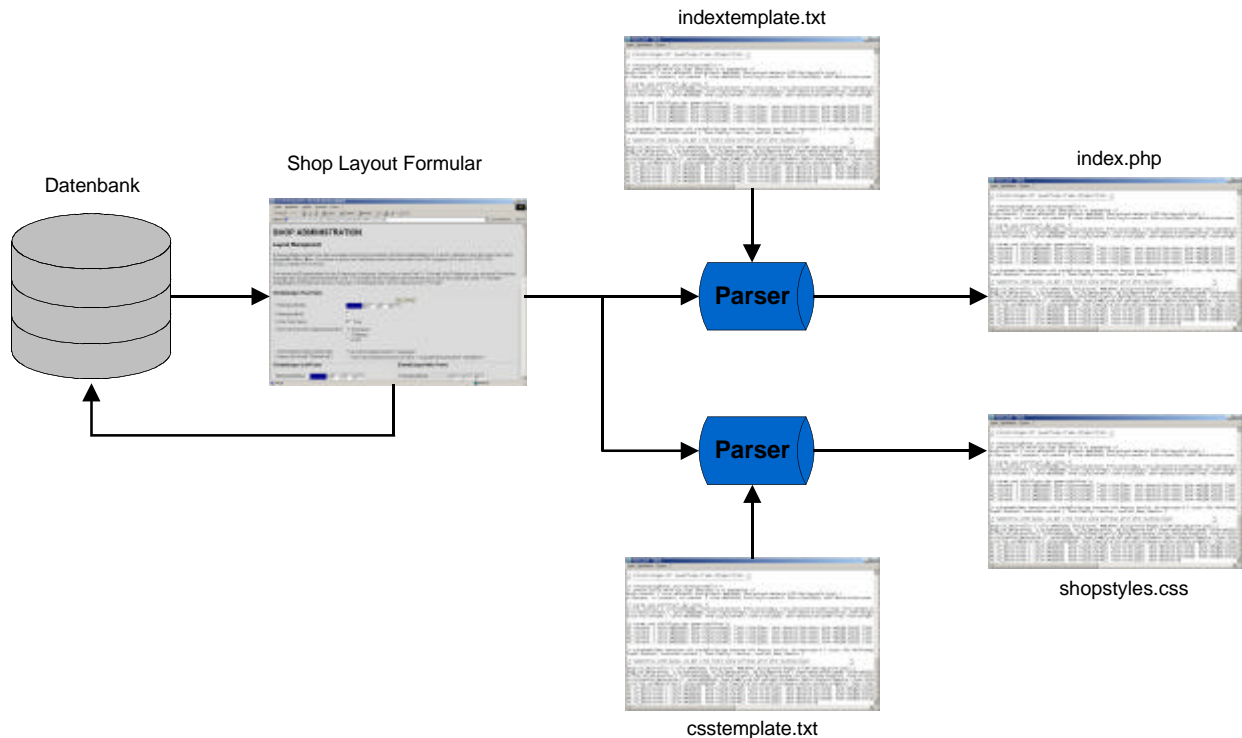
Werte in der Datenbank:

```
left_font_c = #ffffff
left_bg_c = #00008b
left_bg_img = background-image:url(Bilder/bg_left.jpg);
```

Zeile, wie sie in die Datei `shopstyles.css` eingefügt wird:

```
body.left { color:#ffffff; background:#00008b; background-image:
url(Bilder/bg_left.jpg); }
```

Ablauf des Layout Management Vorgangs:



Hintergrundbilder und Shoplogo hochladen

Die Hintergrundbilder und das Shoplogo können mittels Web-Formular hochgeladen werden. Falls der Upload erfolgreich ausgeführt werden konnte, wird das entsprechende Bild im Verzeichnis `shop/Bilder` überschrieben. Da es bei eingeschaltetem SafeMode nicht möglich ist, neue Files im Filesystem zu erzeugen, werden bei der Installation des Shops einige Dummy-Dateien erstellt. Bereits existierende Dateien können auch bei eingeschaltetem SafeMode überschrieben werden.

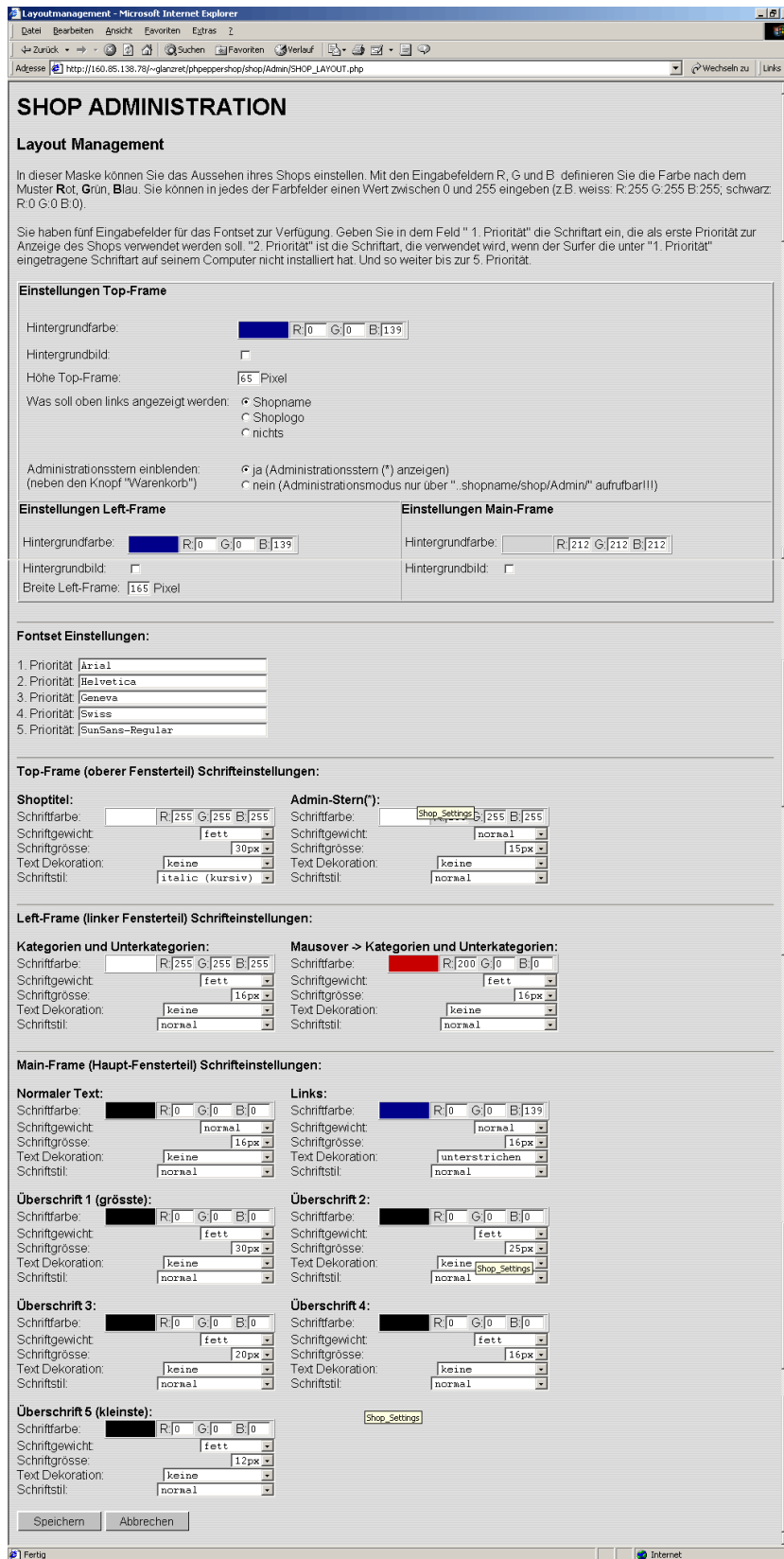
Die akzeptierten Bildtypen zum Upload sind: GIF, JPEG und PNG. Anhand der Dateiendung wird schon vor dem hochladen mittels JavaScript überprüft, ob es sich um einen akzeptierten Bildtyp handelt.

Da man im Layout Management wählen kann, ob ein Hintergrundbild und Shoplogo verwendet werden soll, muss nach erfolgreichem hochladen eines Bildes, die Datenbank aktualisiert werden. Eine Aktualisierung der Datenbank zieht automatisch auch eine Neuerzeugung des CSS-Files nach sich.

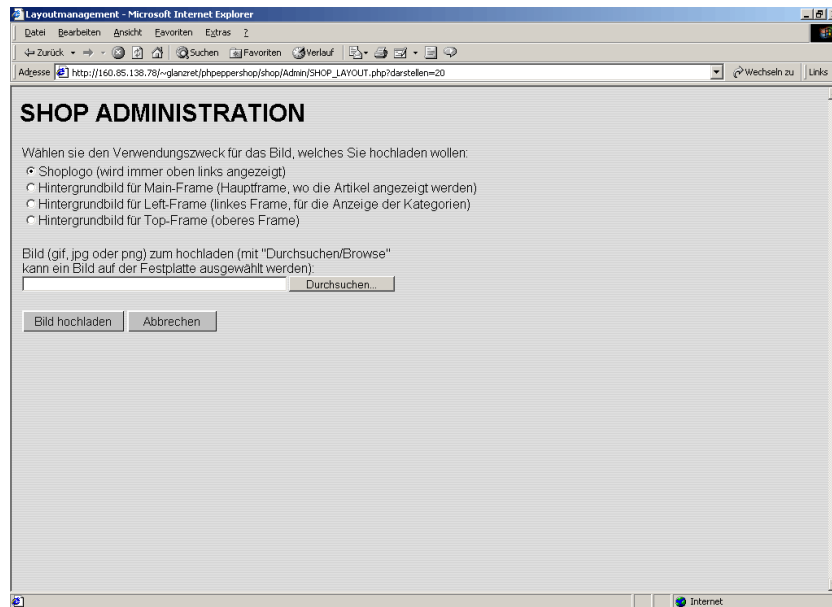
Buttons hochladen

Nach dem gleichen Verfahren, mit dem die Hintergrundbilder hochgeladen werden, können auch die Shopbuttons durch eigene ersetzt werden. Die Buttons müssen vom Typ GIF sein. Hätten wir auch andere Bildtypen zugelassen, wäre das eine ernsthafte Performancegefahr für den Shop gewesen, da der Typ jedes ausgegebenen Buttons in der Datenbank hätte ermittelt werden müssen. Es wären dadurch eine Unmenge von SQL-Abfragen entstanden, die wir unbedingt vermeiden wollten.

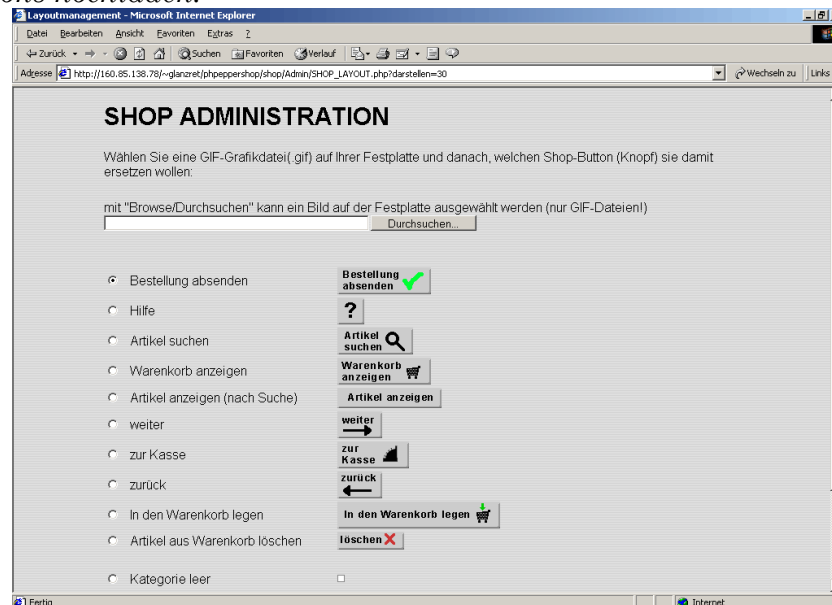
Screenshot des Layout Managements:



Screenshot Bilder hochladen:



Screenshot Buttons hochladen:

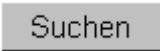





5.4 Buttons im Administrationsbereich

Da nach Abschluss der Diplomarbeit prinzipiell jedermann am PhPepperShop weiterentwickeln kann, werden neue Menüs im Administrationsbereich dazu kommen. Damit trotzdem eine optische Einheitlichkeit bestehen bleibt, haben wir genau definiert, wie Administrations-Buttons auszusehen haben.

Wir verwendeten das Freeware Programm „Buttonz“ von Björn Ischo zur Erzeugung der Buttons. Da die Homepage von Björn Ischo russisch ist, empfehlen wir, das Programm von <http://www.pctip.ch> herunterzuladen.

Die Templates zur Erzeugung der Buttons befinden sich auf der Diplomarbeit-CD. Sie können auch direkt bei uns (developers@phpeppeshop.com) bezogen werden.

<i>Template-File</i>	<i>Abmessungen (b*h)</i>	<i>Beispiel-Button</i>
bt_template_einzeilig_82px.bff	82px * 26px	
bt_template_einzeilig_110px.bff	110px * 26px	
bt_template_einzeilig_125px.bff	125px * 26px	
bt_template_zweizeilig.bff	81px * 32px	

Folgende Namenskonvention sollte beim Erstellen neuer Buttons für den Administrationsbereich eingehalten werden:

bt_ + „Buttonbeschriftung“ + _admin.gif

Wird zum Beispiel ein Button mit der Beschriftung „Backup“ eingeführt, heisst die Datei folgendermassen: bt_backup_admin.gif

5.5 Ändern der Welcome Page

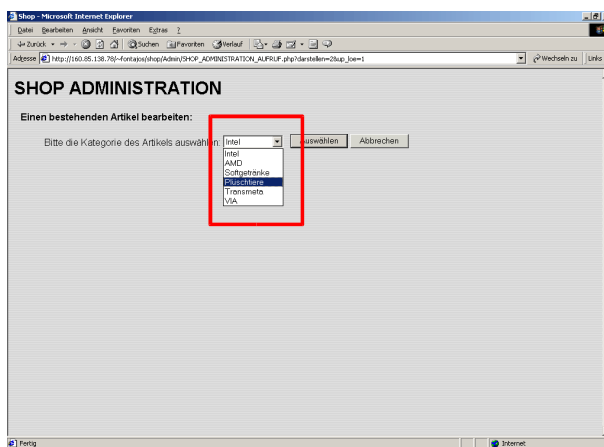
Um die Willkommens-Page abzuändern, muss man nur die Datei content.html im Verzeichnis: <shopdir>/shop/Frameset/ ändern oder ersetzen.

6 Artikelmanagement

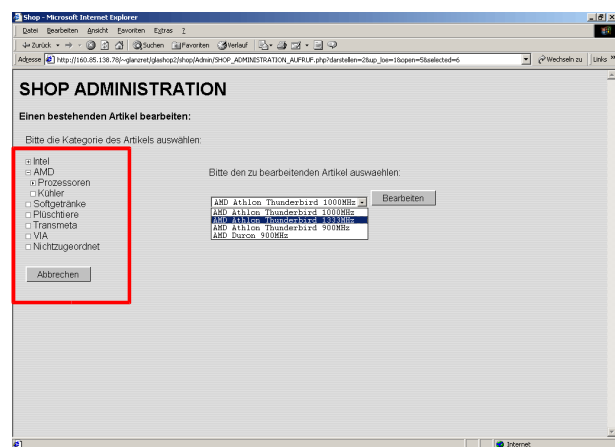
Mit der Einführung von Unterkategorien und der Möglichkeit, Artikel mehreren Kategorien zuzuordnen zu können, waren wir gezwungen, auch das Artikelmanagement gründlich zu überarbeiten. Wir nutzten diesen Umstand, um gleich auch noch die Eingabeüberprüfung mittels PHP durch JavaScript Funktionen zu ersetzen.

Da unser ursprüngliches Menü, zur Auswahl eines Artikels für die Bearbeitung, nicht Unterkategorien tauglich war, mussten wir uns eine andere Lösung zur Auswahl der Kategorie/Unterkategorie des Artikels ausdenken. Das im Left-Frame des Shops verwendete Kategorien-Menü konnte nach einigen Modifikationen sehr gut für diesen Zweck eingesetzt werden. Zur Auswahl des Artikels innerhalb der Kategorie blieben wir bei der bereits in der ersten Version des Shops verwendeten Pull-Down Liste.

Alte Version zur Kategorieauswahl



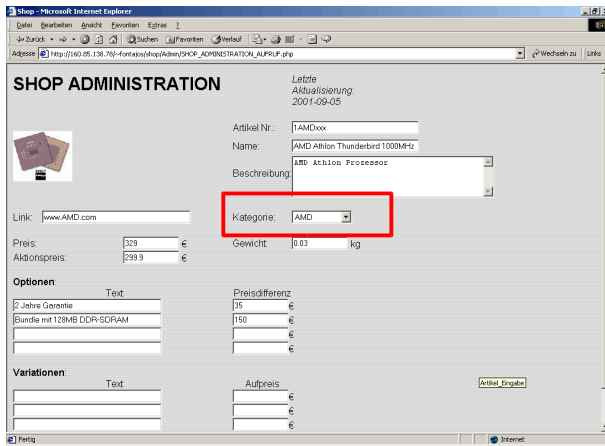
Neue Version zur Kategorieauswahl



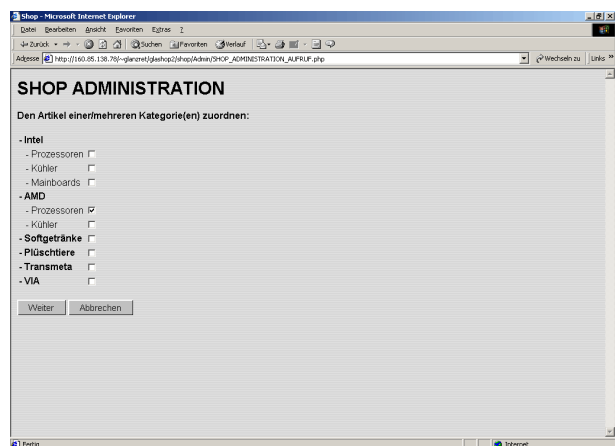
An der Bearbeitung der Artikeldaten änderte sich eigentlich nichts. Da wir die Anzahl Optionen- und Varianten-Felder neu dynamisch konfigurierbar implementierten, mussten wir noch einige Änderungen an der Darstellung des Formulars vornehmen. Die Zuordnung des Artikels zu einer Kategorie konnte nicht mehr im gleichen Formular vorgenommen werden, da der Artikel mit dem neuen Datenbankmodell auch mehreren oder keiner Kategorie zugeordnet werden kann.

Wir mussten also einen weiteren Zwischenschritt einfügen, in dem der Artikel beliebig vielen Kategorien zugeordnet werden konnte. Zu diesem Zweck geben wir die Kategorienliste mit allen Kategorien aufgeklappt aus. Hinter jeder Kategorie und Unterkategorie wird eine Checkbox angezeigt. Soll der Artikel einer Kategorie zugeordnet werden, wird einfach die entsprechende Checkbox angewählt. Enthält eine Kategorie Unterkategorien, können der Kategorie selbst keine Artikel zugeordnet werden.

Alte Version Kategorizuordnung

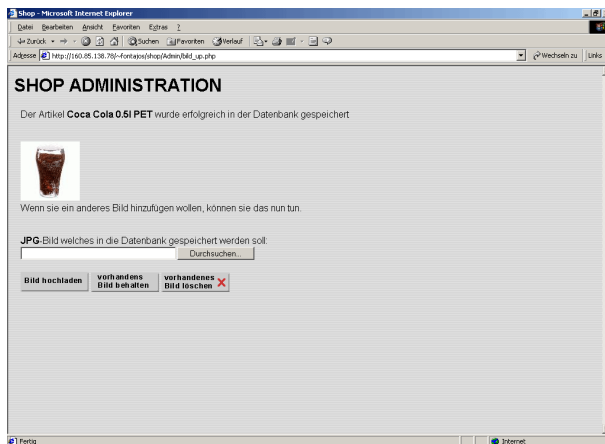


Neue Version Kategorizuordnung

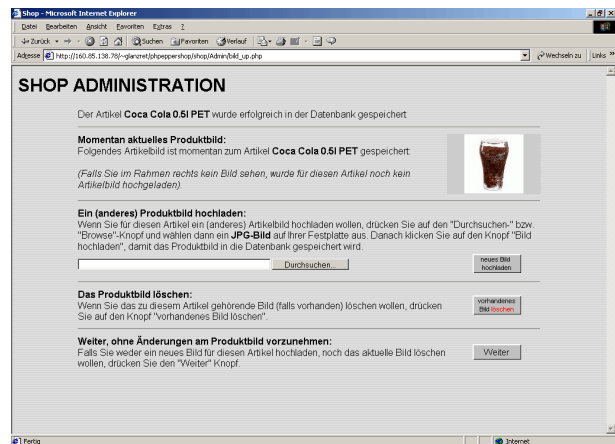


Als letzter Schritt kann für einen Artikel noch ein Bild in die Datenbank gespeichert werden. Der Shopadministrator hat die Wahl, ob er das aktuelle Produktbild löschen, dem Artikel eine neues hinzufügen oder keine Änderungen am Artikel vornehmen will. Das von uns in der Projektarbeit erstellte Formular war aber mehr verwirrend, als hilfreich. Wir wollten, damit die Artikelbearbeitung weiterhin effektiv ausgeführt werden kann, keinen weiteren Zwischenschritt einführen. Deshalb entschlossen wir uns, aussagekräftige Buttons zu erstellen, Hilfetexte direkt in die HTML-Seite auszugeben und die Seite mit horizontalen Querbalken optisch in drei Teile zu gliedern.

Alte Version des Bildmenüs



Neue Version des Bildmenüs



Eine weitere Neuerung ist die Kategorie „Nichtzugeordnet“. Dieser Kategorie enthält alle Artikel, die keiner Kategorie zugeordnet sind. Sie wird im Shop nirgends angezeigt und dient lediglich als Containment für Artikel, die momentan im Shop nicht verwendet werden, oder deren Kategorie bzw. Unterkategorie gelöscht wurde.

Erstellt der Shopadministrator einen neuen Artikel, wird der gleiche Ablauf verwendet, wie bei der Mutation. Per JavaScript wird überprüft, ob die minimal erforderlichen Angaben eingegeben wurden.

An der Funktion für das Löschen eines Artikels aus dem Shop wurde praktisch nichts verändert.

6.1 Artikelsuche

Die Artikelsuche wurde in der Diplomarbeit stark erweitert.

Mehrere Suchbegriffe in einer Abfrage erlaubt

Im Hinblick auf Warensortimente mit vielen ähnlichen Produkten, haben wir die Unterstützung von mehreren Suchbegriffen in einem Suchstring eingeführt. Die Wörter werden, wie von der Suchmaschine Google[®] her gewohnt, AND-verknüpft.

Bilder können optional angezeigt werden

Seitdem wir die SQLs und die Auslesen-Schleife in `getgesuchterArtikel(..)` überarbeitet haben, werden nicht mehr nur der Artikelname und dessen Artikel-ID zurückgegeben, sondern auch gleich noch weitere Attribute. Mit dieser Vorarbeit war es ein Leichtes, die Bilder optional als Suchergebnis anzeigen zu lassen. Damit die Bedienung intuitiv bleibt, wurde das jeweilige Artikelbild auch mit dem Link auf den Artikel versehen.

Mehrfachkategorien Anzeige

Seitdem der PhPepperShop Artikel in mehreren Kategorien gleichzeitig abgelegt werden kann, ist es auch wichtig, in der Suchfunktion des Shops darauf hinzuweisen, wenn ein Artikel in mehreren Kategorien vorhanden ist.

Wir haben uns entschieden nicht nur auf das mehrfache Vorhandensein hinzuweisen, sondern die Mehrfachkategorie-Anzeige gleich mit entsprechenden Links auszustatten. Wir haben dafür die Klasse `Artikelmitkategorien` (`artikel_def.php`) eingeführt und die Funktion `getgesuchterArtikel(..)` entsprechend erweitert. Auf diese Weise kann der Shopbenutzer bequem in die Kategorie springen, in der er den Artikel erwartet hätte.

Beispielsanzeige:

Dieser Artikel ist in mehreren Kategorien vorhanden: [\[Intel -> Prozessoren\]](#) [\[AMD -> Prozessoren\]](#)

Inkrementangabe

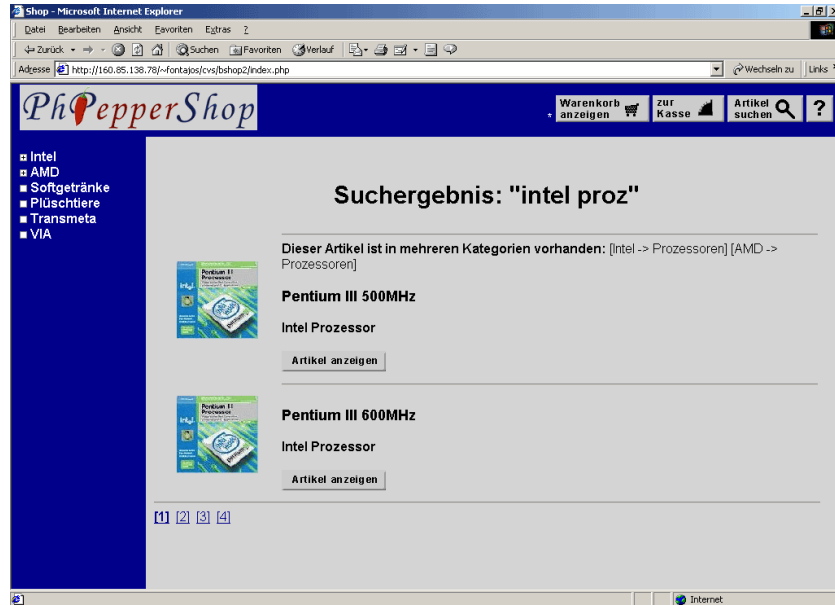
Eine Suche kann sehr aufwändig sein. Es wird eine Query mit mehreren konjunktiv verknüpften LIKE Bedingungen auf die Attribute Name und Beschreibung *aller* Artikel in der Datenbank ausgeführt. Wenn man die Suchbegriffe nicht geschickt auswählt, kann es gut sein, dass man eine enorm grosse Zahl an Treffern erhält. Dies belastet einerseits die Datenbank, sie muss alle Daten auslesen und andererseits die Kundenverbindung, weil alle Daten bis zum Shopbenutzer übertragen werden müssen. Um diesem Problem Herr zu werden, haben wir die Anzahl gleichzeitig angezeigter Suchresultate auf eine in den „allgemeinen Shop-einstellungen“ konfigurierbare Anzahl beschränkt.

Der Funktion `getgesuchterArtikel(..)` wird ein Startwert und eine Anzahl übergeben. Bei der ersten Anzeige ist der Startwert auf 0 gesetzt. Dem Shopbenutzer wird, sobald es mehr Treffer gibt als die konfigurierte maximale Anzahl gleichzeitig angezeigter Artikel, ein Auswahlmenü eingeblendet. In diesem Menü kann er zwischen den Resultat-Intervallen hin und her springen. Beispiel: [\[1\]](#) [\[2\]](#) [\[3\]](#) [\[4\]](#)

Realisiert haben wir diese Funktion mit der LIMIT Angabe. Leider wussten wir nicht, dass LIMIT ein MySQL-Spezifischer Befehl ist und nicht zum SQL'92 Standard gehört. Diese Funktion müsste also noch umprogrammiert werden, sobald man eine andere Datenbank als MySQL einsetzen will.

Screenshot

Hier sieht man ein Suchresultat mit aktivierter Bildanzeige, einem Artikel, welcher in zwei verschiedenen Kategorien eingeteilt ist und einer maximal gleichzeitig angezeigten Anzahl Artikel von zwei:



7 Kategorienmanagement

Das Kategorienmanagement ist sehr umfangreich aufgebaut. Kategorien und Unterkategorien können erstellt, umbenannt, gelöscht oder an eine beliebige Stelle verschoben werden. Entfernt man eine Kategorie, kann man wählen, ob die in ihr enthaltenen Artikel mitgelöscht oder der Kategorie „Nichtzugeordnet“ zugewiesen werden sollen (sofern sie nicht noch einer anderen Kategorie zugeordnet sind).

Wir erstellten gleich zu Beginn der Implementierung ein detailliertes Konzept, welches den Funktionsumfang und Ablauf, sowie die HTML-Ausgabe genau beschreibt. An dieses hielten wir uns konsequent. Da daraus sehr gut die Abläufe zu erkennen sind, wird es hier in *unveränderter Form* abgedruckt.

Im Administrationstool wurden weiter einige Buttons und Texte wurden zwecks besserer Verständlichkeit nochmals überarbeitet (siehe Screenshots weiter unten). Zusätzlich kann beim Löschen einer Kategorie bzw. Unterkategorie noch gewählt werden, ob die in ihr enthaltenen Artikel unwiderruflich gelöscht werden.

7.1 Ursprüngliches Konzept (unverändert)

Es gibt Kategorien und Unterkategorien. Eine Kategorie kann beliebig viele Unterkategorien enthalten.

Funktionen, die mit Kategorien und Unterkategorien ausgeführt werden können:

Kategorie:	- neue Kategorie erstellen	[1]
	- Kategorie verschieben	[2]
	- Kategorie umbenennen	[3]
	- Kategorie löschen	[4]
Unterkategorien:	- neue Unterkategorie erstellen	[5]
	- Unterkategorie innerhalb Kategorie verschieben	[6]
	- Unterkategorie in eine andere Kategorie verschieben	[7]
	- Unterkategorie umbenennen	[8]
	- Unterkategorie löschen	[9]

Kategorienmanagement Bildschirmansicht:

[1]

Prozessoren	[5]	[4]	[2]	[3]
Intel	[6]	[7]	[8]	[9]
AMD	[6]	[7]	[8]	[9]
Motorola	[6]	[7]	[8]	[9]
Gehäuse	[5]	[4]	[2]	[3]
Catan	[6]	[7]	[8]	[9]
Evercase	[6]	[7]	[8]	[9]
Monitore	[5]	[4]	[2]	[3]
Drucker	[5]	[4]	[2]	[3]

neue Kategorie erstellen [1]:

Kategorienname: []

Wo soll die Kategorie eingefügt werden:

hier einfügen

Prozessoren

hier einfügen

Gehäuse

hier einfügen

Monitore

hier einfügen

Drucker

hier einfügen

abbrechen

Kategorie verschieben [2]:

Wohin soll die Kategorie Gehäuse verschoben werden:

hierher verschieben

Prozessoren

hierher verschieben

Monitore

hierher verschieben

Drucker

hierher verschieben

abbrechen

Kategorie umbenennen [3]:

Neuer Kategorienname für die Kategorie Gehäuse: []

ok

abbrechen

Kategorie löschen [4]:

Wenn Sie die Kategorie löschen, werden alle Unterkategorien, die sich darin befinden gelöscht. Artikel, die sich allenfalls noch in dieser Kategorie oder einer ihrer Unterkategorien befinden, werden der Kategorie 'nicht zugeordnete Artikel' zugewiesen.

Wollen Sie die Kategorie Gehäuse mit allen Unterkategorien löschen?

ja

nein

neue Unterkategorie erstellen [5]:

Unterkategorienname: []

Wo soll die Unterkategorie innerhalb der Kategorie Gehäuse eingefügt werden:

hier einfügen

Catan

hier einfügen

Evercase

hier einfügen

abbrechen

Unterkategorie innterhalb Kategorie verschieben [6]:

Wohin soll die Unterkategorie AMD verschoben werden:

hierher verschieben

Intel

hierher verschieben

Motorola

hierher verschieben

abbrechen

Unterkategorie in eine andere Kategorie verschieben [7]:

Wohin soll die Unterkategorie AMD verschoben werden:

Prozessoren

hierher verschieben

Intel

hierher verschieben

AMD

hierher verschieben

Motorola

hierher verschieben

Gehäuse

hierher verschieben

Catan

hierher verschieben

Evercase

hierher verschieben

Monitore

hierher verschieben

Drucker

hierher verschieben

abbrechen

Unterkategorie umbenennen [8]:

Neuer Unterkategorienname für die Unterkategorie Catan:

[]

ok

abbrechen

Unterkategorie löschen [9]:

Wenn Sie die Unterkategorie löschen, werden alle Artikel, die sich noch in der Unterkategorie befinden der Kategorie 'nicht zugeordnete Artikel' zugewiesen.

Wollen Sie die Unterkategorie Catan löschen?

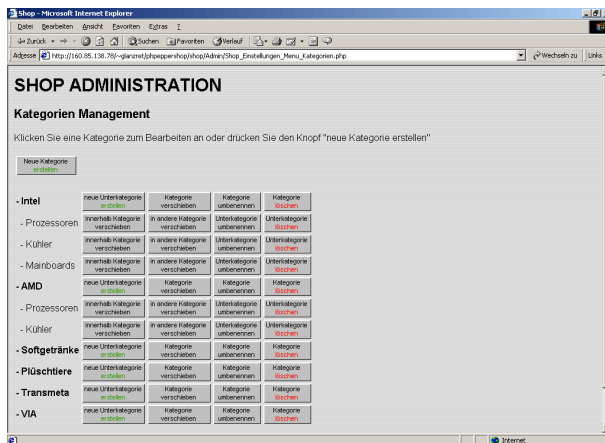
ja

nein

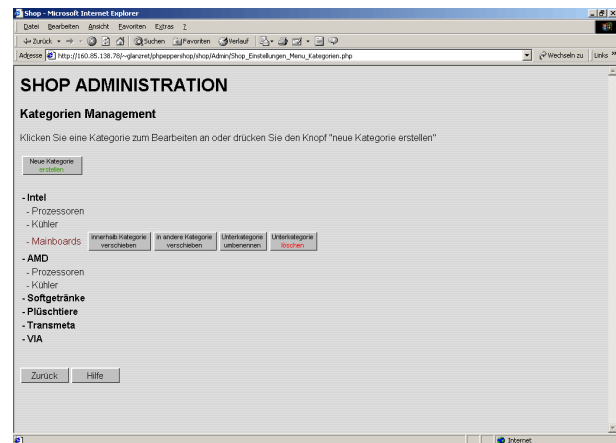
7.2 Hauptmenü

Zu Beginn war geplant, das Hauptmenü des Kategorienmanagements folgendermassen auszugeben: Kategorienliste aufgeklappt mit allen Unterkategorien, rechts davon jeweils die zugehörigen Buttons. Wie im folgenden Bild zu erkennen ist, wurde durch die gegebene Buttonhöhe die Kategorienliste auseinandergezogen. Ausserdem war das Menü durch die grosse Anzahl der Buttons kaum mehr bedienbar.

*schlecht bedienbares
Kategorienmanagement!*



*gut bedienbares
Kategorienmanagement!*



JavaScript bietet glücklicherweise die Möglichkeit, Bilder auf einer ausgegebenen HTML-Seite dynamisch zu ersetzen. Wir entschieden uns dann zuerst für eine sehr dynamische Lösung. Sobald man mit der Maus auf eine Kategorie fährt, sollen gleich neben der Kategorie die Auswahlmöglichkeiten eingeblendet werden. Diese Idee scheiterte leider an der langsamen JavaScript-Engine der Netscape und Mozilla-Browser.

Durch eine kleine Abänderung des JavaScripts von onMouseOver (Aktion ausführen, wenn die Maus über ein Objekt fährt) auf onMouseClick (Aktion ausführen, wenn man mit der Maus auf ein Objekt klickt), entstand dann das schlussendlich verwendete, mit allen Browsern bedienbare Menü. Wird ein Kategorie-, bzw. Unterkategorienname angeklickt, werden rechts davon die für diese Kategorie verfügbaren Auswahlmöglichkeiten eingeblendet. Diese bleiben solange stehen, bis eine andere Kategorie angeklickt wird.

Leider kann mit Netscape 4.7x die Grösse eines einmal ausgegebenen Bildes nicht mehr dynamisch verändert werden. Aus diesem Grund ist eine Browserweiche implementiert, die für Netscape-Browser jeden Button schon bei der Ausgabe des HTML-Dokuments in der vollen Grösse, transparent ausgibt. Da sich dadurch der Abstand der Kategorienamen erhöht, wird Netscape 4.7 Benutzern folgender Hinweis ausgegeben:

Dieser Shopadministrationsmodus ist nicht für Browser der Generation Netscape 4.xx optimiert. Es sind alle Funktionen verfügbar. Die Darstellung ist jedoch nicht optimal!

8 Zahlungsmöglichkeiten

8.1 Vorkasse, Rechnung, Nachnahme und kostenlos

Bei vielen kleineren Shops kommen teure Mailorder Verträge nicht in Frage. Für sie sind die üblichen Zahlungsmethoden wie Vorkasse, Rechnung und Nachnahme die erste Wahl. Mehrere Leute haben uns empfohlen, Vorkasse als Zahlungsmethode zu implementieren. Wir nahmen uns das zu Herzen. Bei Vorkasse ist es praktisch, wenn man die Kontoinformationen gleich mit angeben kann.

Bei der Nachnahmezahlung haben wir noch eine in den Versandkosten getrennt einstellbare Nachnahmegebühr eingeführt. Man kann die Kosten auf diese Weise für den Kunden transparent aufstellen.

Was, wenn ein Kunde nur einen Gratiskatalog bestellt? Wir haben für Bestellungen deren Artikeltotal CHF 0.00 beträgt eine Spezialbehandlung implementiert. Es werden dann keinerlei Versandkosten und Mindermengenzuschläge berechnet.

8.2 Kreditkarten

Wir haben Kreditkarten als Zahlungsmittel implementiert, weil sie allen Unkenrufen des Datenschutzes wegen zum Trotz, eine oft anerkannte und weit verbreitete Zahlungsart darstellen. Weiter kann der Shopadministrator vor der Auslieferung der Bestellung an den Kunden bei den Kreditkartenanbietern verifizieren, ob er sein Geld auch erhalten wird. Diese Garantie hat er bei einer Zahlung auf Rechnung nicht.

Shop-intern abgewickelte Kreditkartenzahlung

Bei einer Shop-internen Kreditkartenzahlung werden die Kreditkartendaten im PhPepperShop erfasst und per E-Mail an den Shopadministrator gesendet. Wir haben diesen Weg der Kreditkartenzahlung implementiert, damit man das günstigste mögliche Kreditkartenhandlung benutzen kann: Es wird *nur* ein Mailorder Vertrag mit dem entsprechenden Kreditkartenanbieter benötigt, mehr nicht. Es entstehen keine weiteren Kosten für Payment Institute wie z.B. yellowpay oder Saferpay. Auf diese Weise können es sich auch kleinere Shops leisten Kreditkarten zu unterstützen.

Einschränkungen

Wir wissen bis jetzt *nur* von Eurocard/Mastercard und VISA, dass sie einen Mailorder Vertrag mit einem Shopbesitzer abschliessen, sobald dieser beweisen kann, dass die Kreditkartendaten nur verschlüsselt übertragen werden. Dies bedingt, dass die Kreditkartendaten bei der Eingabe vom Kunden per SSL verschlüsselt übertragen werden. Weiter muss das E-Mail an den Shopadministrator verschlüsselt werden (z.B. per GNU-PG).

PhPepperShop bietet die (optionale) SSL-Verschlüsselung. Auch wurde der Bestellablauf soweit fertig gestellt, dass man nur noch den E-Mail String verschlüsseln muss (Modul: USER_BESTELLUNG_1.php, Teil: darstellen = 4). Leider konnten wir die Schwierigkeiten die uns PHP's SAFE-MODE bereitete nicht mehr rechtzeitig auf Diplomarbeitsende lösen, so dass dieses Proof of Concept noch nicht integriert wurde. Wir reichen diese Erweiterung aber baldmöglichst nach.

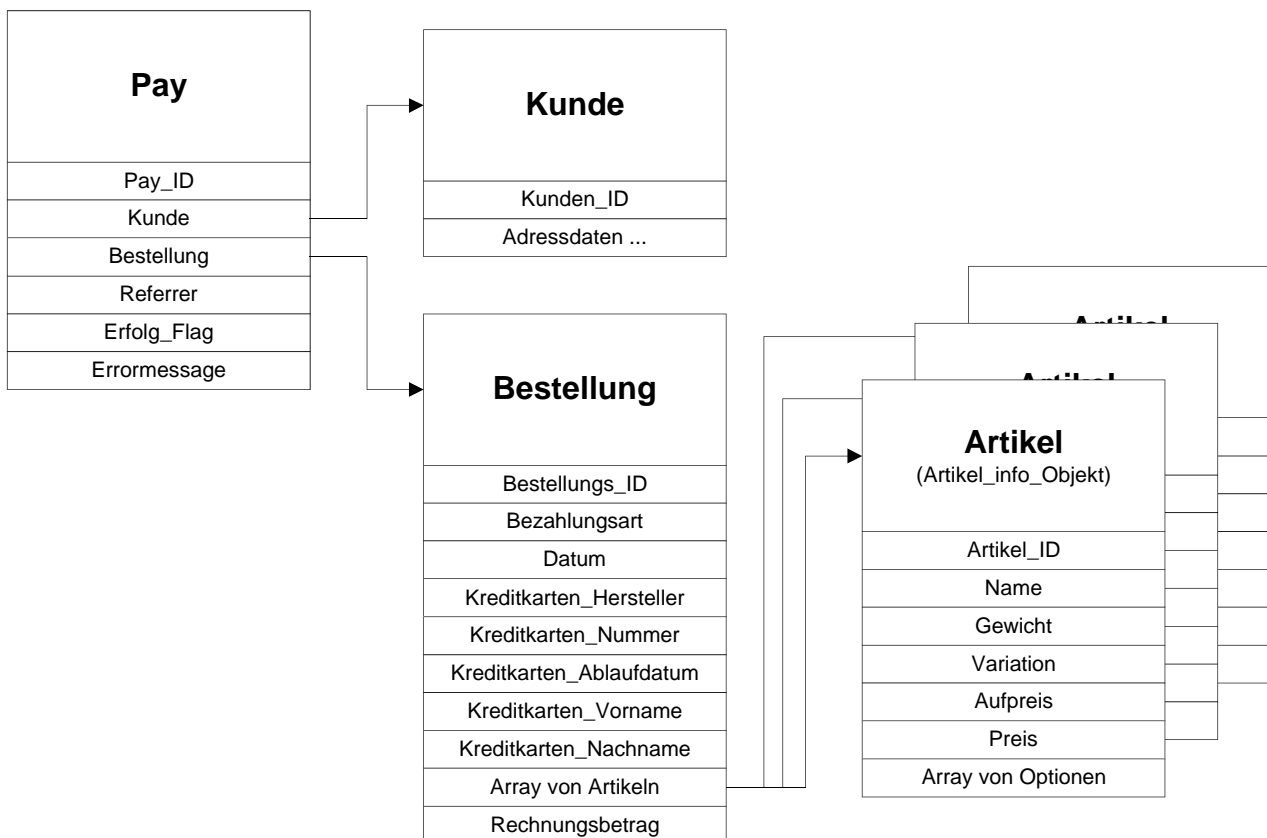
Shop-extern abgewickelte Kreditkartenzahlung

Damit jemand auch die Vorteile eines professionellen Payment Instituts benutzen kann, ha-

ben wir eine Zahlungsschnittstelle eingebaut. Hier kann eine Bestellung an ein externes Zahlungssystem weiter gereicht werden, welches die Zahlung übernimmt. Das externe Zahlungssystem liefert danach eine Meldung über den Erfolg der Zahlung und leitet den Shopbenutzer wieder zu einer vordefinierten Seite im Bestellungsablauf des PhPepperShop zurück. Ein Ablaufdiagramm des Bestellvorgangs kann im Kapitel 1.7.3 (Tests) angeschaut werden.

Idee und Konzept

Die Zahlungsschnittstelle (payment_interface.php) ist eine sehr wichtige Schnittstelle. Man kann mit ihr eine komplette Bestellung bis ins letzte Detail ansehen:



Usability der Schnittstelle

Wir haben einigen Aufwand betrieben um zu vermeiden, dass sich der Benutzer der Schnittstelle mit dem Code des PhPepperShop befassen muss. Dies dient auch der Sicherheit. Es ist immer ein Risiko, wenn jemand am Code des eigentlichen Programms Veränderungen vornehmen muss.

Die Zahlungsschnittstelle befindet sich in der Datei payment_interface.php. Wir haben dort ziemlich detailliert aufgeschrieben, wie man dieses Modul benutzt:

Das Payment Interface definiert ein Formular, welches einem externen Zahlungssystem über hidden-Fields die benötigten Daten zur Verfügung stellt. In der Variable \$pay_here kann man angeben, wo das externe Zahlungssystem die Daten übernimmt.

Das Modul wurde in zwei Teile gegliedert. Im ersten Teil, kann man seine Angaben machen, URL u.s.w., im zweiten Teil (HTML-Formular) muss man nur etwas ändern, wenn man weitere Attribute übergeben will. Wir übertragen per Default alle Kundenattribute und alles was

9 Bestellungsmanagement

Unter einem Bestellungsmanagement verstanden wir die Möglichkeit, offene Bestellungen abzurufen, Bestellungen zu löschen und Kundenadressen (Lieferadressen) zu mutieren - eigentlich eine Verwaltung von offenen Bestellungen.

Wir mussten uns schon früh Gedanken über das Bestellungsmanagement machen, angefangen beim Session Management, bei welchem nach Bestellungsabschluss eines Kunden alle nicht abgeschlossenen, abgelaufenen Sessions gelöscht werden. Hier muss schon überprüft werden, ob auch alle abgeschlossenen Bestellungen von temporären Kunden gelöscht werden. Weiter muss der Shopadministrator das Bestellungsmanagement ein- oder ausschalten können. Es gibt sicher viele kleinere Shops, bei welchen der Shopadministrator die ganze Bestellungsverwaltung über die E-Mails abwickelt.

Da wir auch Manipulationen erlauben wollen (löschen, mutieren), muss die referentielle Integrität von Kunde und Bestellung logisch implementiert werden, denn die Datenbank kann nicht zwei verschiedene Abhängigkeiten mit nur zwei Tabellen nachbilden. Wir haben das auf der Funktionsebene gelöst, indem wir Funktionen zur Kundenbearbeitung, weitere Funktionen zur Bestellungsbearbeitung und wieder andere zum Verändern von Beziehungen zwischen Kunde und Bestellung erstellt haben.

Funktional ist das Bestellungsmanagement fertig ausprogrammiert. Einzig die HTML-Masken zum Bearbeiten der Bestellungen erlauben momentan nur die gleichzeitige Darstellung einer Bestellung mit ihrem Kundendatensatz. Bestellungen können bis jetzt nur aufgrund ihrer Referenznummer angezeigt werden. Dies ist für den Fall gedacht, wenn ein Shopbenutzer nach getätigter Bestellung Änderungen wünscht. Was sicher nächstens implementiert wird, ist die Suche nach Kundennamen und allen abgeschlossenen, noch nicht gelöschten Bestellungen.

10 Kundenattribute Management

Dank dem Kundenattribute Management ist es dem Administrator möglich zu bestimmen, welche Adressinformationen ein Shopkunde eingeben kann und welche von diesen er eingeben muss.

Die Kundenattribut-Bezeichnungen sind in der Tabelle „attribut“ gespeichert. Alle Attribut-Bezeichnungen sind in der Datenbank gespeichert und könnten theoretisch dynamisch verändert werden. Wir haben ein Front-End erstellt, welches 14 fest definierte und vier frei konfigurierbare Attribute enthält. Die fest definierten Attribute sind die Kundenstammdaten.

Die vier Zusatzattribute können vom Shopadministrator frei benannt werden. Für diese kann er wählen, ob sie zum Kunden (z.B. Kundennummer, Geburtsdatum,..) oder zur Bestellung (z.B. gewünschter Liefertermin,..) gespeichert werden.

Alle Attribute können im Shop bei Bedarf darauf getestet werden, ob der Shopkunde das Feld auch ordnungsgemäss ausgefüllt hat. Diese Überprüfung besteht darin, zu testen, ob im entsprechendem Feld etwas ausgefüllt wurde (kein Leerstring). Die Überprüfung der E-Mail-Adresse ist wesentlich umfangreicher und schliesst Fehleingaben nahezu aus.

Da für jedes Attribut gewählt werden kann, ob es verwendet und überprüft werden soll, sind eine riesige Anzahl Kombinationen möglich. Wir mussten also die Ausgabe der Felder, sowie die Überprüfung der gemachten Eingaben dynamisch gestalten.

Zur Überprüfung auf korrekte Eingabe haben wir eine JavaScript Funktion erstellt, die alle Überprüfungsmöglichkeiten beinhaltet:

- keine Überprüfung der Eingabe
- Überprüfung, ob ein Wert eingegeben wurde (kein Leerstring übergeben)
- Überprüfung, ob es sich um eine E-Mail Adresse handelt

Durch den glücklichen Umstand, dass man Formularfeldinhalte eines HTML-Formulars mit JavaScript wie ein Array ansprechen kann, wird ein Zähler laufen gelassen, mit dessen Hilfe jedes Formularfeld auf seine Korrektheit überprüft wird. Damit die Funktion weiss, welches Feld wie zu behandeln ist, wird mit der Ausgabe der HTML-Seite ein JavaScript-Array per PHP erzeugt, der für jedes Feld beinhaltet, wie es überprüft werden soll. Dieser String wird der JavaScript Funktion als Argument übergeben.

Bei der Ausgabe der Bestellung an den Browser, werden die Adressdaten in einer zweispaltigen Tabelle dargestellt. Damit keine Löcher, durch nicht verwendete Adressattribute entstehen, wird auch diese Tabelle dynamisch erzeugt. Ist schlussendlich nur noch ein Adresselement auf der letzten Zeile, wird diese durch eine Dummy-Zelle ergänzt.

11 CRM – Customer Relationship Management

Customer Relationship Management ist ein beliebig dehnbarer Begriff und momentan ein Modewort. Wir haben in unserem Shop ein einfaches Customer Relationship Management.

Von der Feedback Analyse (Kapitel 1.4) her, teilten wir die Customer Relationship Management Vorschläge und unsere eigenen Ideen in zwei Kategorien ein:

CRM 1: Eigenes Login

- Automatisch ausgefüllte Adresse
- Login zwingend / nicht zwingend --> Händlermodus

CRM 2: Persönlichen Ansprechpartner konfigurieren können

- Rechnung
- Lieferschein drucken können
- Bestellungsfortschritt abrufbar (versendet, ...)

Das CRM2 mussten wir nach einer ersten Aufwandschätzung aufgrund seiner Komplexität und den benötigten weiteren Funktionen weglassen. Wir schauten uns also das CRM 1-Paket genauer an. Nach ersten Designentscheiden haben wir herausgefunden, dass wir den Händlermodus nur implementieren könnten, wenn wir eine PHP-Authentifizierung an die Session binden würden. PHP-Auth der PHPLib wäre ein Weg um dies zu implementieren. Wir haben also auch hier gesehen, dass der Aufwand nicht zu unterschätzen ist und haben uns daraufhin dafür entschieden, vorerst nur die Grundelemente des CRM 1-Pakets zu implementieren.

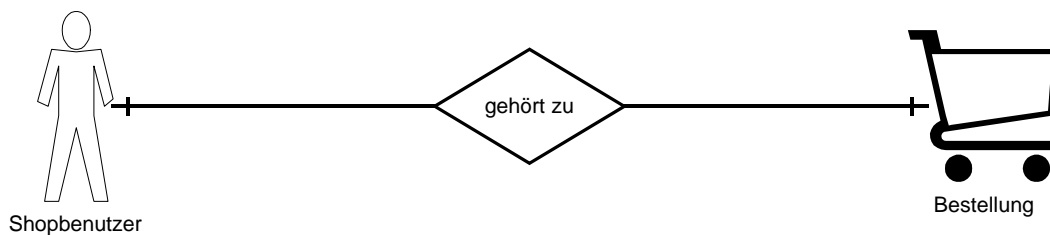
11.1 Welche Funktionen bietet das CRM

Wir wollten dem Kunden die Möglichkeit geben, seinen Kundendatensatz persistent zu speichern. Diese Grundfunktionalität erforderte viele weitere Funktionen und Anpassungen.

- Kunde kann sich einloggen oder auch nicht (freie Wahl).
- Kunde kann sein Passwort vergessen und es sich per E-Mail senden lassen.
- Kunde wird nach einmaligem Login wieder erkannt, bis er seinen Browser schliesst.
- Kunde kann seine Daten verändern (Datenbank-Update).

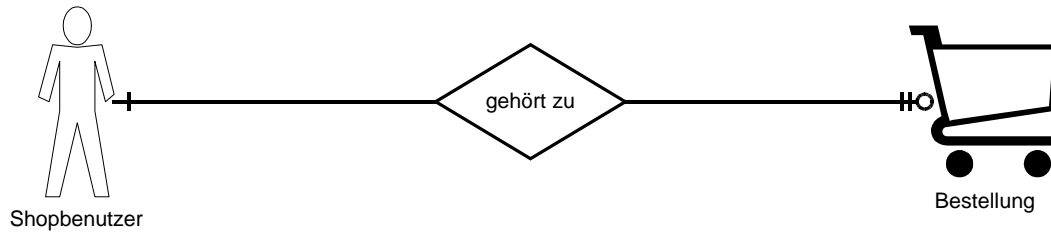
Ein detailliertes Ablaufdiagramm des CRM's findet man im Kapitel 1.7.3 (Tests).

Ein weiteres Problem war, dass es durch das Einführen des CRM's persistente und temporäre Kunden gibt. Wenn ein Kunde sich nicht beim Shop anmeldet, muss sein Kundendatensatz nach der Bestellung wieder gelöscht werden, aber nur dann, wenn das Bestellungsmanagement ausgeschaltet ist. Wenn man das Bestellungsmanagement einschaltet, gibt es bei einem temporären Kunden eine Beziehung zu seiner Bestellung: Ein temporärer Kunde ist an eine Bestellung gebunden und kann nur maximal eine Bestellung haben:



Bei persistenten Kunden hingegen ist die Bestellung an einen Kunden gebunden. Wenn man

hier eine Bestellung löscht, soll natürlich nicht auch der Kunde mit gelöscht werden. Ausserdem kann ein persistenter Kunde mehrere Bestellungen haben, nicht nur eine. Zum Schluss muss noch beachtet werden, dass bei Bestellungsabschluss und ausgeschaltetem Bestellungsmanagement nur die Bestellung, nicht aber der Kunde selbst, gelöscht werden darf.



Man sieht also, dass schon die grundlegenden CRM-Funktionen einiges an Analyseaufwand benötigten, um richtig implementiert zu werden.

Der PhPepperShop bietet dem Kunden jetzt an, seine Adressdaten automatisch ausfüllen zu lassen und das sogar noch SSL geschützt, wenn der Shopbetreiber SSL einschaltet.

12. Anhang

12.1 Literaturverzeichnis

Nachfolgend eine Liste mit allen Quellen und Literaturverweisen, aufgliedert nach den einzelnen Themen, die durch das Webshop-Programming tangiert werden.

12.1.1 PHP

PHP 4 Grundlagen und Profiwissen

Carl Hanser Verlag München Wien
Jörg Krause
November 2000
1161 Seiten
ISBN 3-446-21546-8

Dies war, wie auch in der Projektarbeit, unser 'Hauptbuch'. Es enthält eine Vielfalt an Themen, die alle ziemlich ausführlich erklärt werden. Aber auch dieses Buch hat seine Grenzen (Bilder in Datenbank speichern, grössere Projekte mit PHP)

PHP Grundlagen und Profiwissen

Carl Hanser Verlag München Wien
Jörg Krause
März 2000
1067 Seiten
ISBN 3-446-21301-5

Dieses Buch ist die Vorgänger-Version des oben genannten. Wir haben es selten konsultiert, aber doch ab und zu etwas daraus brauchen können.

PHP kurz & gut

O'Reilly Verlag Köln
Rasmus Lerdorf
1. Auflage 2000
111 Seiten
ISBN 3-89721-225-0

Eine praktische, kompakte Zusammenfassung, aber leider nur für PHP3.

PHP 4 Das bhv Taschenbuch

bhv Verlag Landsberg
Uwe Hess, Günther Karl
1. Auflage April 2001
764 Seiten
ISBN 3-8287-5057-5

Eine gute Einführung in die Grundlagen von PHP4, der Stoff geht aber nicht all zu tief.

Linux Enterprise 'PHP Special'-Ausgabe

Sonderausgabe Herbst 2001
98 Seiten

Ein Blick in die aktuelle PHP Entwicklung, mit Vorstellung der PHPLib und PEAR. Viele interessante, professionelle Ansätze.

PHP Archive / -Foren

<http://www.hotscripts.com/>
<http://www.phpbuilder.com/>
<http://www.phparchiv.de/>

Diese drei Pages halfen uns während der Entwicklung oft weiter. Viele Scripts findet man auf hotscripts und im phparchiv. Kompetente Foren in phpbuilder.

12.1.2 Apache Webserver

Apache Web-Server Installation, Konfiguration, Administration

MITP-Verlag GmbH Bonn

Lars Eilebrecht

3. Auflage 2000

671 Seiten

ISBN 3-8266-0612-4

Ein sehr tief gehendes Buch. Half uns sehr gut bei der Apache-Konfiguration und beim Einrichten von htaccess.

12.1.3 CRM – Customer Relationship Management

iX Magazin für professionelle Informationstechnik

Ausgabe 8/2001, ab Seite 38

Operatives und analytisches CRM, CRM und Betriebswirtschaft

Ausgabe 7/2001, ab Seite 104

Dienen und Verdienen - CRM: Der Wert eines Kunden

Wir haben umfassende und sehr aktuelle Artikel in der renommierten Zeitschrift iX gefunden.

Die Artikel geben dem Modewort CRM etwas Farbe. Man lernt, dass CRM nicht einfach eine Technik oder eine Sammlung von Funktionen ist, sondern eine Unternehmensphilosophie. Man muss, um ein umfangreiches CRM einzuführen, weit mehr machen, als es in einen Shop zu integrieren.

Heinz Heise Verlag, 30604 Hannover

12.1.4 Datenbanken und im speziellen MySQL

MySQL in 21 Tagen

Markt + Technik Verlag München

Mark Maslakowski

2001

580 Seiten

ISBN 3-8272-5850-2

Dies war unser 'Hauptbuch' im Bereich Datenbanken. Man findet auch knifflige Aspekte von MySQL hier drin. Vor allem für die MySQL Konfiguration sehr zu empfehlen.

SQL, der Schlüssel zu relationalen Datenbanken

Rowohlt Taschenbuch Verlag GmbH Reineck bei Hamburg

Gregor Kuhlmann, Friedrich Müllmerstadt

Mai 1999

318 Seiten

ISBN 3-499-60063-3

Dieses Buch half uns vor allem bei schwierigen Queries, aber auch bei Fragen, wie man gewisse Datenbank-Aktionen überhaupt erst angehen soll. Sehr zu empfehlen.

Datenbanken – Vorlesung an der Zürcher Hochschule Winterthur

Dr. Hans-Walter Buff, zweites Jahr IT

Der Stoff half uns beim Entity Relationship Design, aber auch bei den Create-, Insert- und Del-SQL-Scripts.

12.1.5 HTML, JavaScript und Cascading Style Sheets

HTML 4 – Die Sprache des Web

dpunkt-Verlag Heidelberg

Robert Tolksdorf

3. Auflage 1997

308 Seiten

ISBN 3-920993-77-2

Bietet einen tiefgehenden Einblick ins HTML, leider schon etwas in die Jahre gekommen.

Jetzt lerne ich HTML

Markt + Technik Verlag München

Harald Taglinger

1997

290 Seiten

ISBN 3-8272-5305-5

Gutes Nachschlagewerk (Farbtabelle, ...) immer wieder praktisch für 'wie ging das jetzt schon wieder?'-Momente. Leider auch schon etwas in die Jahre gekommen.

Das grosse Buch JavaScript

DATA BECKER GmbH & Co KG Düsseldorf

Rainer Kolbeck

1. Auflage 1997

422 Seiten

ISBN 3-8158-1321-2

Dieses Buch ist unbrauchbar (zumindest wenn man HTML schon kennt und den Netscape Javascript Debugger nicht erklärt braucht...)

JavaScript

verlag moderne industrie, 53227 Bonn

Michael Seeboerger-Weichselbaum

1. Auflage 2001

767 Seiten

ISBN 3-8266-8106-1

Wir konnten dieses Buch sehr gut als Nachschlagewerk gebrauchen, da es eine umfassende und logisch aufgebaute JavaScript Befehlsreferenz beinhaltet.

Wir können es nur weiter empfehlen, das Preis- / Leistungsverhältnis stimmt.

Einführung in JavaScript

<http://atlas.ee.fhm.edu/mm-cd/skript/javascript/>

Eine wirklich gelungene Einführung in JavaScript. Vermittelt die Grundlagen von JavaScript in nur 20 Minuten.

HTML-Referenz

<http://www.teamone.de/selfhtml/>

Selfhtml ist und bleibt das beste Nachschlagewerk in Sachen HTML. Sogar für JavaScript konnten wir einige Informationen von dieser Adresse gebrauchen. Sehr zu empfehlen.

12.1.6 Perl

Einführung in die Perl-Programmierung

<http://userpage.fu-berlin.de/~corff/perl/perl-kurs.html>

Dies ist ein guter Einstieg und trotzdem eine sehr umfangreiche Page. Wir haben vor allem das Kapitel 2, Grundlegende Elemente der Sprache, verwendet.

Perl – Ein 25 Minuten Crashkurs!

http://www.opensecure.de/PerlansB/perl_crashkurs.htm

Eine nette Einführung. Auf nur 20 Seiten hat man so ziemlich alle Sprachelemente, die man zu Beginn braucht, aufgelistet.

Reguläre Ausdrücke in Perl

<http://www.oreilly.de/catalog/regexger/chapter/kap-2d.html>

<http://www.heise.de/ix/artikel/1998/11/178/01.shtml>

Details zur Muster Erkennung mit Regulären Ausdrücken mit Perl.

12.2 Software Versionen**12.2.1 Auf dem Server eingesetzte Software**

SuSE Linux 7.1

Kernel: 2.4.0-4GB (i686)

Apache

Version 1.3.20 (Unix) (LAMPS Installation)

MySQL

Version 3.23.35

PHP4

Version: 4.0.6

phpMyAdmin

Version 2.2.0

Einzige Linux-Distribution, die zu Projektbeginn schon den Kernel 2.4.0 beinhaltet.

In SuSE Linux Distribution enthalten, genügend aktuell.

Die zu Projektbeginn aktuellste MySQL Datenbank Version aus dem Internet.

Diese damals aktuellste Version von PHP4 war glücklicherweise schon in der SuSE Distribution enthalten.

Datenbank Administrationstool mit Webinterface. Dieses Programm brauchten wir, um SQL-Queries auszutesten und Änderungen in der Datenbank manuell vorzunehmen.

12.2.2 Auf den Arbeitsstationen eingesetzte Software

Windows 2000 Professionell

Komfortable Entwicklungsoberfläche

SP1

Apache für Windows

Zu Testzwecken eingerichtet

Version 1.3.19

PHP4

Zu Testzwecken eingerichtet

Version 4.04pl1

PHPed

Sehr nützlicher PHP-Editor mit integriertem FTP-Tool, welches direktes Arbeiten auf dem Server ermöglicht. Leider noch etwas Beta.

Version 2.96.1.2

CUTE FTP

FTP-Client

Version 4.2.3 Build 2.14.1

Putty

Telnet / SSH Client

Version 0.5.1

12.3 Zusatzdokumente

Auf der Homepage des PhPepperShop Projekts (www.phpeppershop.com) findet man unter Dokumente sogenannte Zusammenfassungen. Dies sind Dokumente welche wir während unserer Projekt- und Diplomarbeit erstellten.