

Hochschule Ostwestfalen-Lippe
Fachbereich 8 - Umweltingenieurwesen und Angewandte Informatik
Fachgebiet Angewandte Informatik
Projekt Realisierung betrieblicher IT-Systeme
4. Semester SoSe 2015

**Ausarbeitung:
Projekt Realisierung betrieblicher IT-Systeme**

**Entwicklung eines Webshops
für das Fahrradproduktions- und Vertriebsunternehmen
Global Bike Inc.**

Von
Stefan Schnürer,
Benedikt Brüntrup,
Raphael Gregarek und
Michael Dück

Erstprüfer: Prof. Dr. rer. nat. Stefan Wolf
Zweitprüfer: Prof. Dr. Ralf Hesse
Drittprüfer: Prof. Dr. Jessica Rubart
Vierprüfer: Prof. Dr. rer. nat. Burkhard Wrenger
Eingereicht am: 22. August 2015

Inhaltsverzeichnis

1	Projektaufgabenstellung	2
2	Projektplanung	2
2.1	Ideenfindung	2
2.2	Anforderungsanalyse	3
2.3	Geschäftsprozess	3
2.4	Risikoanalyse	5
2.5	Wahl der Programmiersprache	5
2.6	Wahl der Programmierentwicklungsumgebung und des Versions-verwaltungs- Plugin	5
2.7	Grundlegender Aufbau und Ordnerstruktur der Webseite	5
2.8	Die Test-Server	7
3	Arbeitspakete	8
3.1	Einteilung der Arbeitspakete	8
3.2	Arbeitspaket von Herrn Schnürer	9
3.3	Arbeitspakete von Herrn Brüntrup	15
3.3.1	Arbeitspaket 5: Die Schnittstellendatenbank	15
3.3.2	Arbeitspaket 6: Registrierungs- und Anmeldungsfenster designen und programmieren	21

1 Projektaufgabenstellung

Die Aufgabenstellung des Projektes ist es einen Webshop für die Firma Global Bike Inc. (folgend GBI) zu erstellen. Diese Firma stellt Fahrräder her und vermarktet dieses. Die Produkte des Webshops sollen aus einem ERP-System geladen werden. Bei dem ERP-System handelt es sich um ein schon bestehendes SAP-ERP-System. Alle auf dem Webshop getätigten Bestellungen sollen an das ERP-System weitergereicht werden. Zudem soll für den Benutzer des Webshops der Stand der Bestellung einsehbar sein. Dieser Stand soll ebenfalls aus dem ERP-System übernommen werden.

Weitere Kriterien für den Webshop sind, dass dieser in eine APP für mobile Geräte einbindbar sein muss, ein Corporate-Design entworfen werden, sowie ein Marketing-Mix für die Online-Vermarktung im Großraum OWL entwickelt werden soll.

2 Projektplanung

Dieser Abschnitt erläutert, wie die Projektplanung dieses Projekts ablief. Es wird der Ablauf von der Ideenfindung bis durch Planung der Durchführung erläutert.

2.1 Ideenfindung

Der erste Schritt des Projekts bestand darin zu bestimmen, was für Funktionen der Webshop alles anbieten soll. Hierfür sind drei Onlinewebshops betrachtet worden. Es wurden die Webshops „fahrrad.de“, „amazon.de“ und „alternate.de“ betrachtet.

Vom Webshop „alternate.de“ ist der Webseitenkopf als sehr sinnvoll empfunden worden. Dieser Kopf ist am oberen Rand der Webseite fest fixiert und wird nicht mitgescrollt. Dieses bietet den Vorteil, dass der Kunde durchgehend die Menge einsehen kann, die sich im Warenkorb befindet. Grund hierfür ist, dass sich eine kleine Warenkorbanzeige mit im Kopf der Webseite befindet. Zudem kann der Benutzer sich durch die Webseite navigieren ohne zu scrollen. Da ebenso im Kopf der Webseite die Suche fixiert ist, kann der Benutzer auch ohne zu scrollen nach etwas suchen. Wegen den vielen Vorteilen wurde beschlossen den Kopf der Webseite weitgehend für den Webshop der GBI zu übernehmen.

Aus allen drei Webshops wurde zudem übernommen, dass der Webshop der GBI mit einen Warenkorb gelöst werden soll und die Benutzer sollen Kommentare zu einen Artikel schreiben und lesen können. Ebenso soll es möglich sein, dass der Kunde den Artikel bewerten kann. Der Artikel soll über eine Suche und eine Navigation zugreifbar sein. Zu einen Artikel soll in der Produktbeschreibung mehrere Produktbilder angezeigt werden, die sich der Kunde durch draufklicken größer ansehen kann. Die Produktbeschreibungsseite soll mindestens den Preis, die Bewertung, den Produktnamen und eine ausführlichen Beschreibungstext erhalten.

Als negativ wurde empfunden, wenn die Startseite der Webseite „überladen“ von Produkten ist. Somit wurde beschlossen die Startseite des Webshops übersichtlich zu gestalten.

2.2 Anforderungsanalyse

Der zweite Schritt des Projekts war die Anforderungsanalyse. Hier wurde ermittelt, welche Anforderungen genau der Webshop erfüllen muss. Während der Analyse einigte man sich auf folgende Projektanforderungen:

- Man sollte sich registrieren können.
- Wenn man schon registriert ist, sollte man sich anmelden können.
- Die Daten zwischen dem ERP-System und dem Webshop sollen über eine Schnittstellendatenbank ausgetauscht werden.
- Um auf mobilen Geräten und Desktop-Computern die Webseite richtig anzeigen zu können soll eine zusätzliche CSS-Datei für mobile Geräte erstellt werden. Diese Datei soll die Style-Anweisungen überschreiben, die vom mobilen Gerät falsch angezeigt werden.
- Man sollte ein Artikel über die Navigation und die Suche erreichen.
- Die gefundenen Artikel sollten in einen Warenkorb gesammelt werden können (mit Mengenangabe).
- Nachdem der Warenkorb gefüllt ist, sollen die Artikel bestellt werden können. Hierbei sollen verschiedene Liefer- und Zahlungsmöglichkeiten auswählbar sein.
- Die Bestellung sollte mit einer Bestätigungs-Email bestätigt werden.
- Der Zustand der Bestellung sollte nachverfolgbar sein.
- Der Kunde sollte den Artikel bewerten können und Kommentare äußern können.

2.3 Geschäftsprozess

Als nächstes wurde festgelegt, wie ein Geschäftsprozess beim Webshop aussehen soll. Der Geschäftsprozess beginnt, wenn der Kunde die Webseite aufruft. Darauf folgend registriert sich entweder der Kunde oder er logt sich mit der vorhandenen Benutzerkennung ein. Danach sucht sich der Kunde Produkte auf der Webseite aus und füllt damit den Warenkorb. Nachdem der Warenkorb gefüllt ist, führt der Kunde die Bestellung durch. Wenn die Firma GBI die Bestellung erhalten hat, wird an den Kunden eine Bestätigungsmail versendet. Darauf folgend wartet der Webshop auf die Bezahlung der Produkte. Hat die Firma GBI die Zahlung erhalten, wird die Ware an den Kunden verschickt.

Dieser Geschäftsprozess ist nochmal in dem als „Business Process Model and Notation“ dargestellten Geschäftsmodell (*Abbildung 1*) zu sehen.

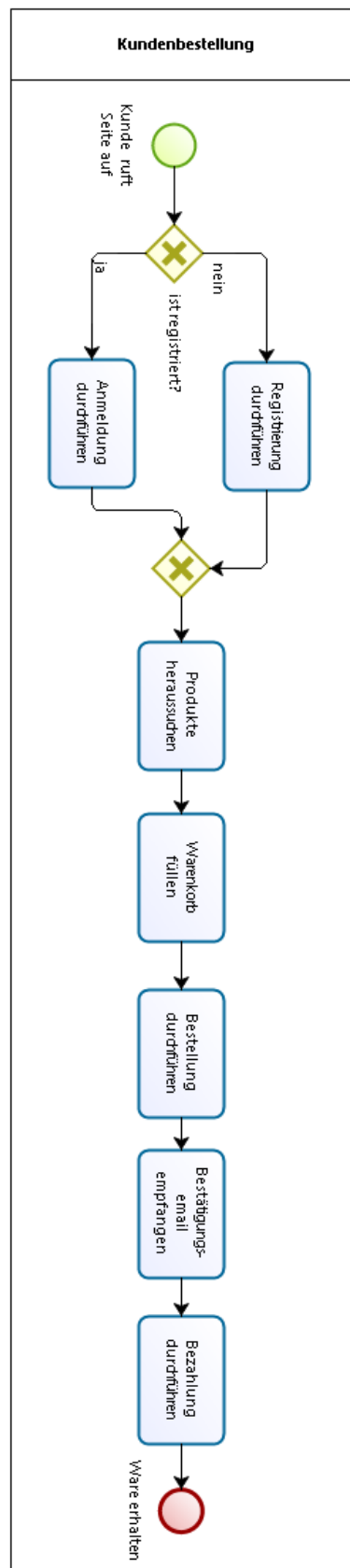


Abbildung 1: Geschäftsprozess des Webshops

2.4 Risikoanalyse

Platzhalter für die Risikoanalyse

2.5 Wahl der Programmiersprache

Nachdem die Anforderungen an das Projekt festgelegt waren, wurde sich mit der Festlegung der Programmiersprache beschäftigt. Es stand Java Enterprise Edition und PHP zur Auswahl. Die Auswahl fiel auf PHP, da aus alten Projekten und der Webdesign-Vorlesung schon Vorwissen über die Programmierung von Webseiten in PHP vorhanden war. Zudem war auch geplant Teile des Quellcodes der alten Projekte für das neue Projekt wieder zu benutzen.

2.6 Wahl der Programmierentwicklungsumgebung und des Versionsverwaltungs-Plugin

Zum Beginn des Projekts hat jedes Gruppenmitglied einen eigenen Texteditor und an einer eigenen Version des Projekts gearbeitet. Es fiel nach kurzer Zeit auf, dass so das Zusammenfügen der verschiedenen Versionen sehr aufwendig war. Aus diesem Grund wurde nach einer Möglichkeit gesucht, das Problem zu lösen. Es wurde das „GitHub-Plugin“ für das Programm „Eclipse“ als Lösung für das Versionsproblem gefunden.

Als Entwicklungsumgebung wurde sich für Eclipse entschlossen, da es bei installierten PHP-Plugin eine gute Autovervollständigung für PHP bietet. Zudem ist es als Portable-Version erhältlich und kann somit ohne Installation auf den Hochschulrechnern betrieben werden.

Als Versionsverwaltungssystem wurde sich für „GitHub“ entschlossen, da dieses schon in den Programm „Eclipse“ integriert ist. Zudem ist „GitHub“ eine Cloud-basierte Lösung, welche das gemeinsame Arbeiten an einem Projekt ermöglicht. Das komplette Projekt wird dabei auf einen Server hochgeladen. Jedes Gruppenmitglied, das einen Account bei „GitHub“ erstellt hat, kann dem Projekt beitreten, indem es bei den Gruppen-Administratoren um Erlaubnis bittet. Dadurch kann ein Dokument von mehreren Gruppenmitgliedern gemeinsam bearbeitet werden. So entstehen mehrere Versionen von einem Dokument, die anschließend wieder mit Hilfe eines in GitHub enthaltenen „Merge-Tools“ zu einer gemeinsamen Version zusammengefügt werden können.

2.7 Grundlegender Aufbau und Ordnerstruktur der Webseite

Einer der ersten Schritte der praktischen Umsetzung des Projekts war die Festlegung der Ordnerstruktur der Webseite. Die Ordnerstruktur sollte einen einheitlichen Aufbau der Webseite garantieren. Mit der Festlegung sollte für jedes Projektmitglied definiert werden, wo welche Dateien abgelegt werden sollen und wie generell die programmiertechnische Struktur des Programms aussehen soll.

Als Programmieransatz für das Projekt wurde die objektorientierte Programmierung gewählt. Die Hauptgründe für die Auswahl des Programmieransatzes waren, dass die objektorientierte Programmierung durch die Vererbung die Möglichkeit bietet, schon programmierte Funktionen an andere Klassen zu übernehmen. Es wird also Programmierarbeit eingespart.

Ein Beispiel für die Einsparung von Programmierarbeit ist, wenn ein Fehler in einer vererbten Methode ist, muss der Fehler nur bei der Eltern-Klasse behoben werden und nicht bei der Kind-Klasse. Ein weiterer Grund ist, dass die objektorientierte Programmierung auch die Übersichtlichkeit der Webseite fördert. Für jedes Objekt der Webseite wird einfach eine Klasse angelegt, welche dieses definiert. So wird z.B. eine Klasse für den Kunden oder den Warenkorb erstellt. Diese Klasse arbeitet alle Aufgaben des Objektes ab. Der letzte Grund für den Ansatz war, dass schon objektorientierte Quellcodes aus alten Projekten zur Verfügung standen.

Da sich für die objektorientierte Programmierung entschieden wurde, hat sich die Gruppe auf folgende Ordnerstruktur geeinigt:

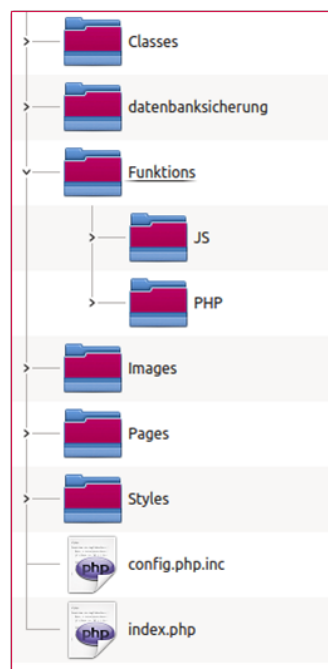


Abbildung 2: Ordnerstruktur des Projekts

Der Ordner „Classes“ beinhaltet alle PHP-Klasse des Projekts. Dieser Ordner enthält zum Beispiel Klassen zur Umsetzung des Warenkorbs, des Datenbankzugriffs, des Kunden, der Kundenbestellung und so weiter. Der Ordner „datenbanksicherung“ enthält die jeweils aktuelle Sicherung der Datenbank. Der folgende Ordner ist in zwei Unterordner aufgeteilt. Dieser Ordner wurde mit dem Namen „Funktionen“ bezeichnet. Der erste Unterordner, der den Namen „JS“ trägt, enthält die clientseitig programmierten Funktionen. Die clientseitige Programmierung sollte in JavaScript umgesetzt werden. Aus diesen Grund sind in diesen Ordner JavaScript-Dateien vorzufinden. Die JavaScripts werden unter anderen dafür verwendet eine Formulareingabe vor den Absenden auf Gültigkeit zu überprüfen. Dieses sorgt dafür, dass es nicht zu unnötigen Datenverkehr zwischen Browser und Webserver kommt. Der Unterordner „PHP“ enthält die Funktionen, die zum Beispiel nach dem Absenden einer Formulareingabe an den Webserver aufgerufen werden. Diese Funktionen instanziierten die Klassen. Zudem rufen die Funktionen-Dateien die entsprechenden Methoden auf, um die von Formular empfangenen Eingaben zu verarbeiten. Dieser Ordner enthält noch zwei weitere Dateien. Die

erste Datei der beiden heißt „set_page.php.inc“. Diese Datei liest einen über die Adressenzeile erhaltenden Parameter aus. An Hand des Parameters bestimmt die Datei, welcher Inhalt aktuell auf der Webseite angezeigt werden soll. Hierzu inkludiert die Datei die entsprechende HTML-Seite aus den Ordner „Pages“. Die zweite Datei heißt „set_control.php.inc“. Diese Datei inkludiert die benötigte Funktions-Datei, die z.B. die Formulareingaben bearbeiten. In diesen Projekt werden keine in PHP programmierten Funktionen und Inhaltseiten direkt aufgerufen. Diese Dateien werden immer in die Index.php inkludiert. Der Grund hierfür ist, dass so den Anwender der interne Aufbau der Webseite verborgen bleibt.

Der Ordner „Images“ enthält die Bilder, die auf der Webseite angezeigt werden. Der letzte Ordner mit den Namen „Styles“ enthält die CSS-Dateien, die das Design der Webseite festlegen. In der „config.php.inc“-Datei werden grundlegende Konfigurationen für den Webshop festgelegt. In dieser Datei wird unter anderen die Zugangsdaten für die Datenbank angegeben. Die „index.php“ ist die standardmäßige Webseiteneinstiegsdatei. Über diese Datei wird die Webseite geladen.

2.8 Die Test-Server

Um bei der Entwicklung nicht erst darauf warten zu müssen bis die Server-Gruppe die benötigten Server-Dienst aufgesetzt hat, wurde sich nach eine Möglichkeit umgesehen, wie man für Testzwecke die Webseite auf den Hochschulrechnern laufen lassen könnte. Hierfür wurde das Programm XAMPP gefunden. Es ist eine Software-Sammlung, die alle benötigten Server-Dienste für PHP-Webseiten auf einen Windows-Rechner bereitstellt. Die Softwaresammlung bietet den Vorteil, dass eine portable Version erhältlich ist. Diese Version kann ohne Administrationsrechte ausgeführt werden.

3 Arbeitspakete

Dieser Abschnitt beschreibt die Unterteilung des Projektes in Arbeitspakete und wie die Gruppenmitglieder die zugeteilten Arbeitspakete abarbeitet haben. Es wird auch auf den unvorhergesehenen Arbeitsbereitschaftsmangel der Gruppenmitglieder Herr Gregarek und Herr Dück eingegangen.

3.1 Einteilung der Arbeitspakete

Zunächst wurden die Arbeitspakete aus der Anforderungsanalyse abgeleitet. Aus dieser Ableitung sind folgende Arbeitspakete definiert worden:

1. Entwicklung des grundlegenden Webseitendesigns für Desktop-Rechner
2. Anpassung des Designs für mobile Geräte
3. Entwicklung der Webseitennavigationsleiste
4. Entwicklung der inhaltlichen Seiten des Webshops:
 - Startseite
 - Produktseite
 - Rechtliche Seiten:
 - ◊ Impressum
 - ◊ Datenschutz
 - ◊ AGB
5. Entwicklung einer Schnittstellendatenbank zum Datenaustausch zwischen Webshop und SAP-System:
 - Ansprechpartner der SAP-Gruppe für die Schnittstelle zum Webshop
 - Entwurf von Tabellen für die Datenhaltung des Webshops
6. Registrierungs- und Anmeldungsfenster designen und programmieren
 - Möglichkeit den Kunden bieten die Anmeldeinformationen später verändern zu können
7. Entwickeln einer Suchfunktion:
 - Autovervollständigung
 - Erweiterte Suchfunktion, wo die Suche genauer eingegrenzt werden kann
 - Auflistung der Suchergebnisse
8. Warenkorb
 - Auflistung der Produkte, die sich im Warenkorb befinden

- Formular mit Mengenfeld, um ein Produkt in den Warenkorb zu packen
- Möglichkeit zur nachträglichen Änderung der Produkte im Warenkorb:
 - ◊ Änderung der Bestellmenge
 - ◊ Entfernen des Produkts aus dem Warenkorb

9. Bestellvorgang:

- Bestellroutine mit folgenden Schritten:
 - ◊ Wahl der Versandart
 - ◊ Nachträglichen Änderung der Bestellmenge der Produkte
 - ◊ Wahl Zahlungsart
 - ◊ Wahl Lieferadresse
 - ◊ Bestellübersichtseite zur Kontrolle vor den Absenden der Bestellung
- Einsicht des aktuellen Status der Bestellung
- Möglichkeit die Bestellung zu stornieren

10. Erstellung eines Marketing Mix

11. Bewertungsfunktion von Artikel (mit Kommentarfunktion)

12. Email-Versand

- Klasse für den Email-Versand entwerfen
- Designen der Bestätigungs-E-mails

Die oben genannten Arbeitspakete hat sich die Gruppe untereinander aufgeteilt. Stefan Schnürer hat Arbeitspakete 1,3 und 4 übernommen. Benedikt Brüntrup übernahm die Arbeitspakete 5, 6 und 7. Michael Dück hatte sich bereit erklärt die Arbeitspakete 2, 11 und 12 zu bearbeiten. Raphael wollte die Arbeitspakete 8, 9 und 10 abarbeiten.

3.2 Arbeitspaket von Herrn Schnürer

In den nachfolgenden Wochen wird die Homepage nach Vorlage des Designs Entwurfs von Herrn Brüntrup erstellt. Der erste Prototyp der Seite basiert auf den Standardaufbau einer Webseite. Dabei wird der Inhalt der Seite in HTML geschrieben, während eine CSS Datei für das Seitenlayout der Webseite verantwortlich ist. Abbildung 1 zeigt einen ersten groben Designentwurf der Startseite:

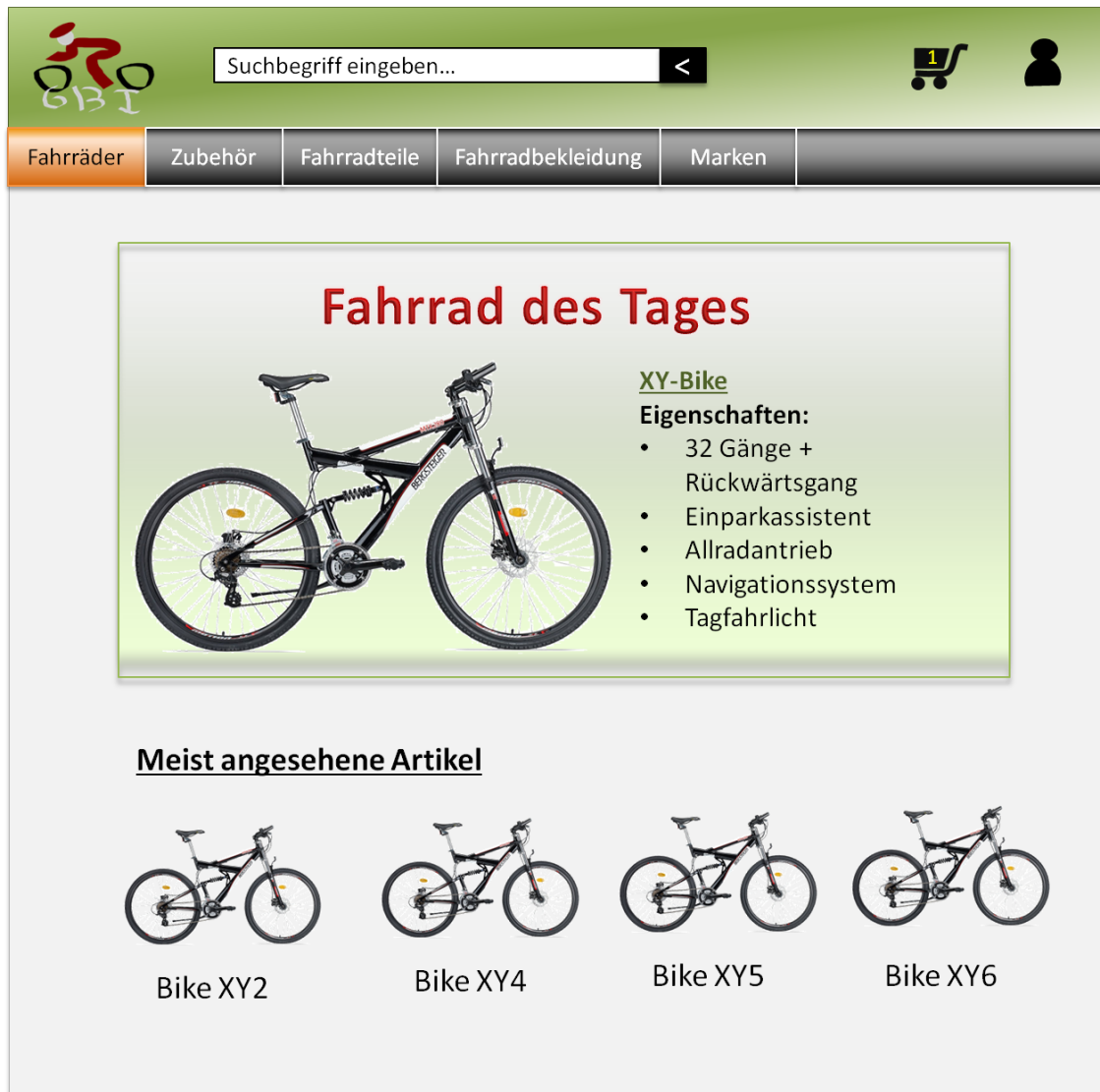


Abbildung 3: Erster grober Designentwurf der Startseite

Die Startseite wird zunächst mit zwei einfachen Artikeln versehen. Sie sollen eine grobe Vorstellung liefern, wie die fertige Startseite aussehen könnte. In den darauf folgenden Wochen liegt der Fokus auf die Seitennavigation in der Kopfzeile (nachfolgend Header) der Seite. Die Seitennavigation wird mithilfe von CSS an den Designentwurf angepasst. Es werden die Hauptkategorien „Fahrräder“, „Zubehör“, „Fahrradteile“, „Fahrradbekleidung“, „Marken“ und eine Rubrik „HowTo“ angezeigt. Letztere verlinkt auf eine Internetquelle, mit dessen Hilfe die Seitennavigation umgesetzt wurde. Diese Kategorie dient den anderen Gruppenmitgliedern als Referenz und wird in späteren Versionen des Webshops wieder entfernt. Fährt der Nutzer mit der Maus über die Kategorie „Fahrräder“ öffnet sich ein Drop-Down-Menü, welches den Nutzer verschiedene Fahrradmodelle auswählen lässt. Die komplette Seitennavigation wurde in CSS3 umgesetzt. CSS3 kann von nahezu jeden aktuellen Browser gelesen werden und bietet zudem den Vorteil, dass der entsprechende Code in der bisherigen CSS-Datei eingefügt

werden kann. Da CSS3 unabhängig von Java Script ist, lässt sich die Seite zudem problemlos bedienen, wenn Java Script auf den jeweiligen Browsern deaktiviert ist. Dies begründet die Entscheidung CSS3 für die Seitennavigation zu verwenden. Abbildung 2 und 3 zeigen den bisherigen Seitenprototypen:

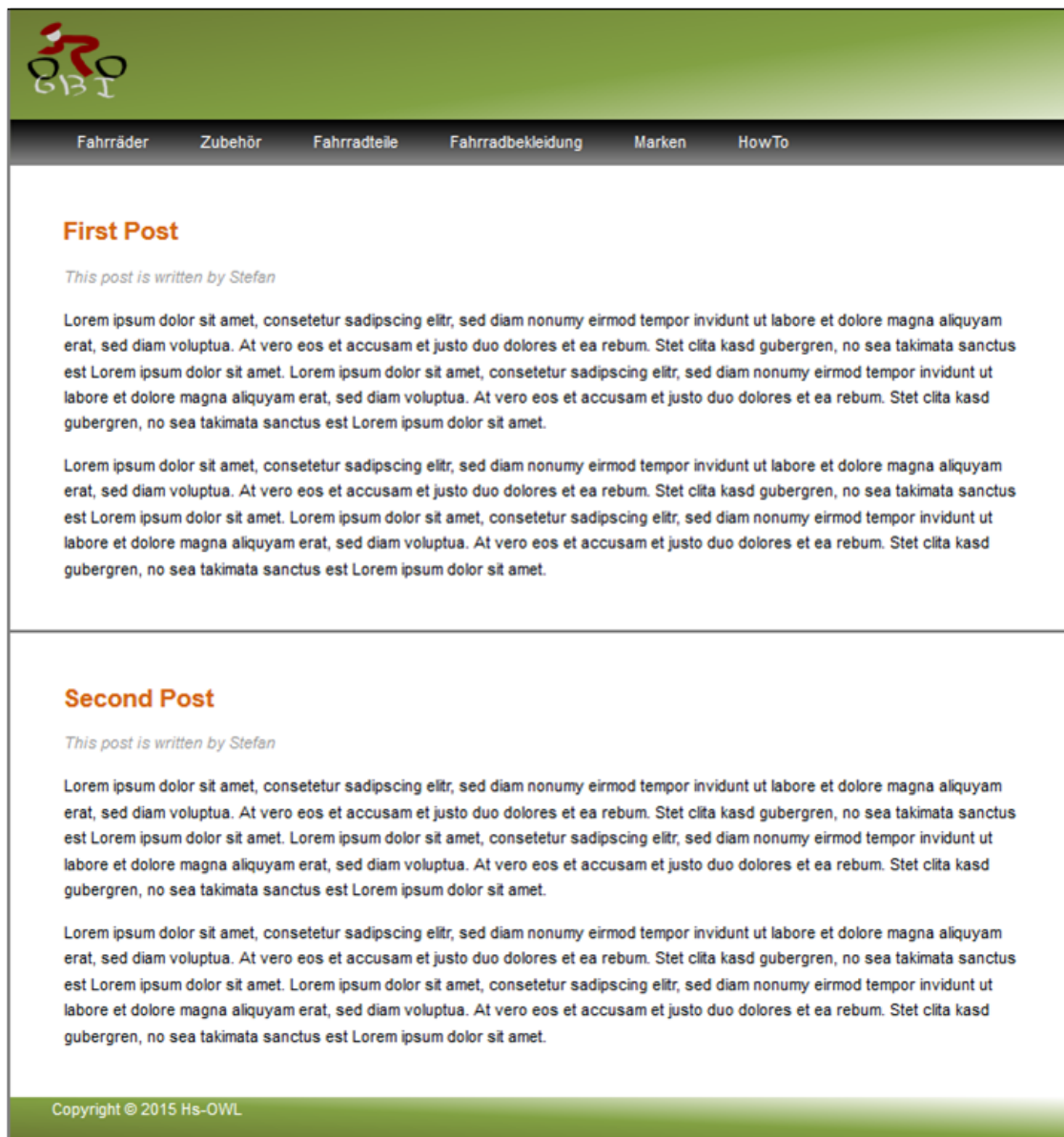


Abbildung 4: Seitenprototyp



Abbildung 5: Seitenprototyp mit Drop-Down-Menü

Im Laufe der nächsten Woche wird die Startseite des Webshops gemäß dem Designentwurfs erstellt (siehe Abbildung 4). Auf der Startseite ist das „Angebot des Tages“ zu sehen. Es gibt eine Kurzbeschreibung zu dem Artikel mit den wichtigsten Eigenschaften. Der untere Teil der Startseite zeigt die meist angesehenen Artikel des Nutzers an.

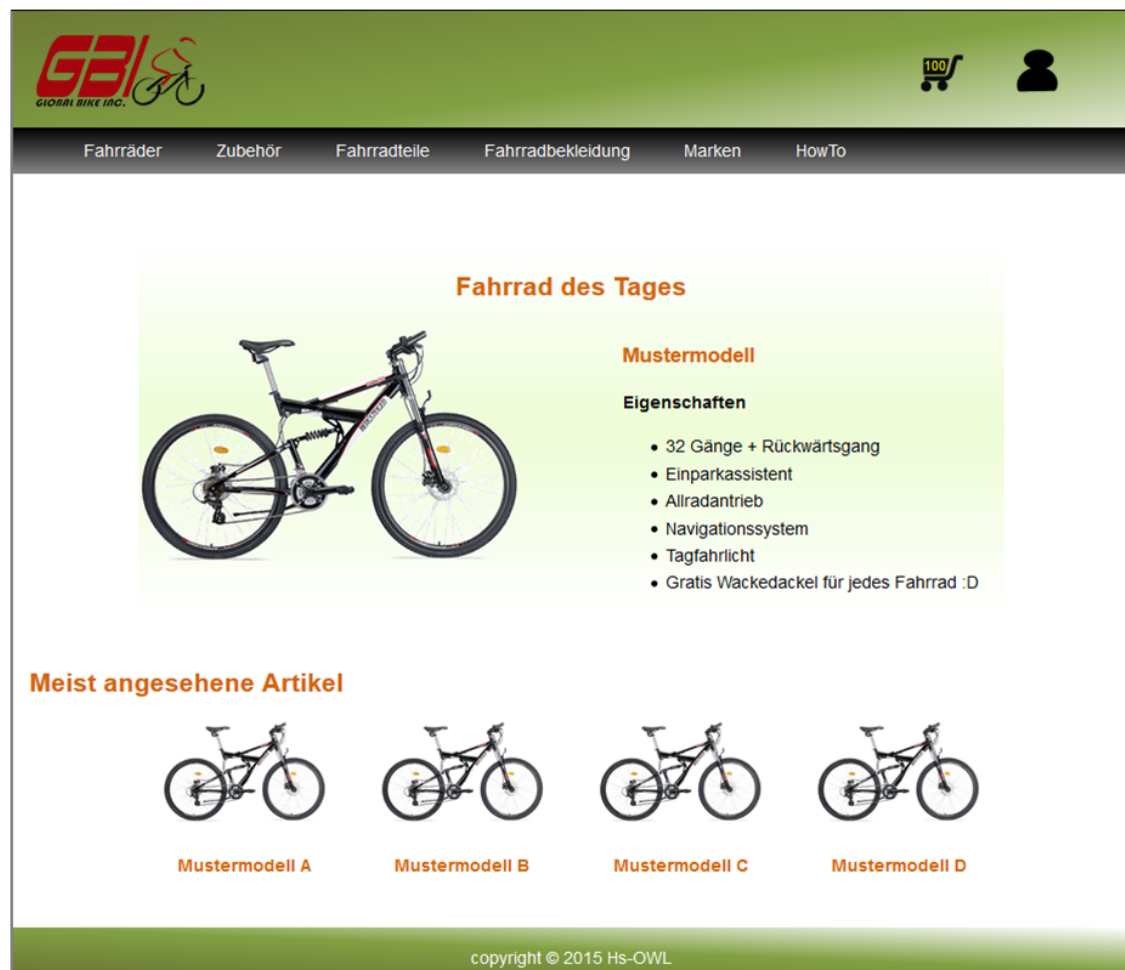


Abbildung 6: Startseite des Webshops

In den nächsten Tagen wird das Thema Rechtsschutz behandelt. Dem Webshop werden ein Impressum, die AGB sowie Hinweise zum Datenschutz hinzugefügt:

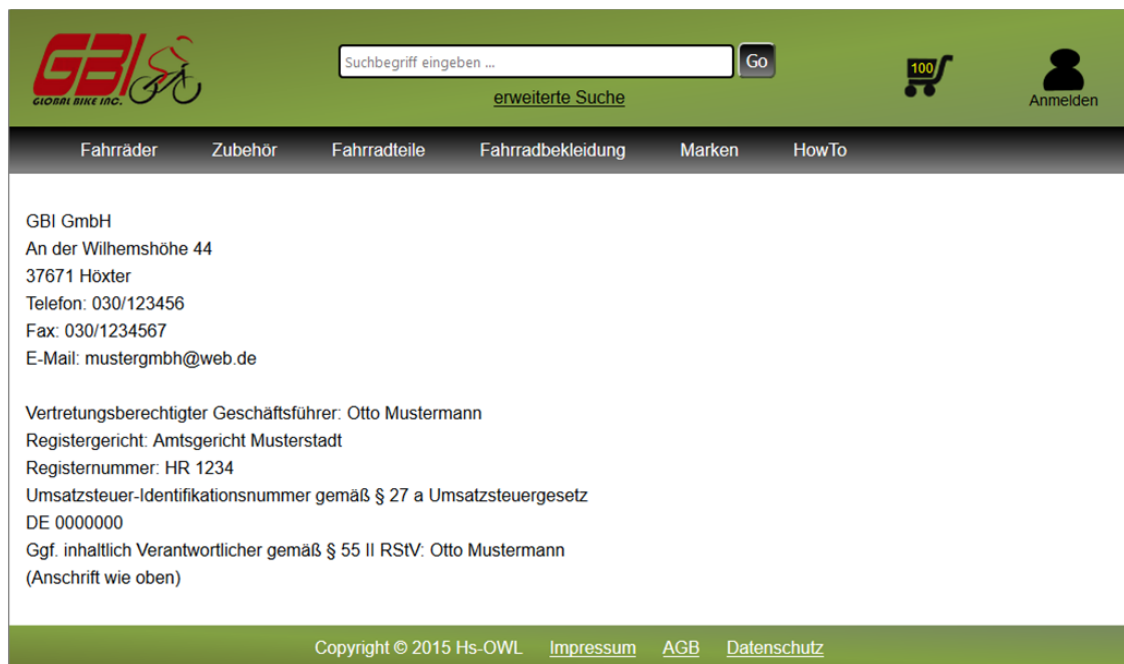


Abbildung 7: Impressum des Webshops



Abbildung 8: AGB des Webshops



Abbildung 9: Datenschutz des Webshops

3.3 Arbeitspakete von Herrn Brüntrup

In diesen Abschnitt wird beschrieben, wie Herr Brüntrup seine Arbeitspakete gelöst hat. Zudem wird in diesen Abschnitt noch beschrieben welche Arbeitspakete wegen Zeitmangel und fehlender Arbeitsbereitschaft zusätzlich noch übernommen wurden.

3.3.1 Arbeitspaket 5: Die Schnittstellendatenbank

Die Datenbank stellt die Schnittstelle zwischen den SAP-System und den Webshop da. Aus diesem Grund wurde die Datenbankstruktur in der ersten Woche nach der Arbeitspaketverteilung zusammen mit Mitgliedern der SAP-Gruppe entworfen.

Gründe für die Schnittstellendatenbank

Die Gründe warum die Kommunikation mit einer Schnittstellendatenbank gelöst wurde wird in den folgenden Sätzen genannt. Der erste Grund war die Verfügbarkeit. Der Webshop sollte auch erreichbar sein, wenn mal keine Verbindung zum SAP-System besteht. Der SAP-Server steht extern bei einer Hochschule in Magdeburg. Es könnte also schon mal dazu kommen, dass das System nicht erreichbar ist. Ein anderer Grund war die Flexibilität. Die Datenerhaltung sollte flexible erweiterbar sein. Soll die Webseite erweitert werden, z.B. durch eine

360°-Ansicht des Fahrrads, kann einfach die benötigten Felder zur Datenbank hinzugefügt werden. Beim SAP-System ist das nur beschränkt möglich. Der Zugriff auf das SAP-System wird mit vordefinierten Funktionen gelöst. Diese Funktionen werden „Business Application Programming Interface (BAPI)“ genannt. Da die SAP-Gruppe beim SAP-System keine Berechtigungen hat, neue BAPIs zu erstellen wäre somit die Erweiterbarkeit des Webshops sehr eingeschränkt.

Aufbau der Schnittstellendatenbank

Nun wird der zu Beginn geplante Aufbau der Schnittstellendatenbank erläutert. Die zu Beginn geplante Datenbank bestand aus den Relationen „Kunde“, „Bestellungen“, „Produkte“ und „Produktbilder“. Die Relation „Kunde“ enthält die Login-Daten des Kunden, sowie dessen Adresse. Die Relation „Bestellungen“ enthält alle Angaben zur Bestellung. Angaben sind z.B. die Zahlungs- und Versandart, sowie der Status der Bestellung. Die vorletzte genannte Relation beinhaltet alle Angaben zu den Produkten, die beim Webshop verkauft werden sollen. Ihr wurde deswegen auch den Namen „Produkte“ gegeben. Die Relation „Produktbilder“ enthält Pfadangaben, wo auf den Server die Produktbilder abgelegt sind. Da zu einem Produkt mehrere Produktbilder abgelegt werden können, wurde sich auf eine extra Tabelle für die Produktbilder geeinigt.

Eine Relation, die die Produkte einer Bestellung enthält, wurde zunächst vergessen umzusetzen. Das Vergessen der Relation fiel erst auf, als die SAP-Gruppe mit der Umsetzung des Bestellvorgangs bei der SAP-Schnittstelle beginnen wollte. Zur Lösung des Problems fügte die SAP-Gruppe nun nachträglich eine Relation „Bestellprodukte“ zur Datenbank hinzu. Diese Relation steht in Beziehung mit den Relationen „Bestellungen“ und „Produkte“. Die Relation beinhaltet die Produkte einer Bestellung.

Als Datenbank-Server wurde sich auf MySQL geeinigt. Dieser Server lässt sich problemlos auf Linux installieren. Es war wichtig, dass die Datenbank unter Linux läuft, da die Server-Gruppe einen Linux-Server aufsetzen wollte. Zudem bietet MySQL den Vorteil, dass auf eine MySQL-Datenbank mit der Programmiersprache PHP gut zugegriffen werden kann. Die SAP-Gruppe wollte die Datenbank mit einer selbst geschriebenen JAVA-Anwendung füllen. Auch von JAVA aus kann gut auf eine MySQL-Datenbank zugegriffen werden. Ein weiterer Grund für die Nutzung der MySQL-Datenbank war, dass die Softwaresammlung XAMPP einen MySQL-Server beinhaltet. Die Softwaresammlung XAMPP wurde von der Webshop-Gruppe als Testserver zum Testen der Webseite auf den Hochschulrechnern verwendet. Die *Abbildung 10* zeigt das „Entity-Relationship-Diagramm (ERD)“. Dieses Diagramm ist die erste Version der Datenbank, die zusammen mit der SAP-Gruppe entworfen wurde.

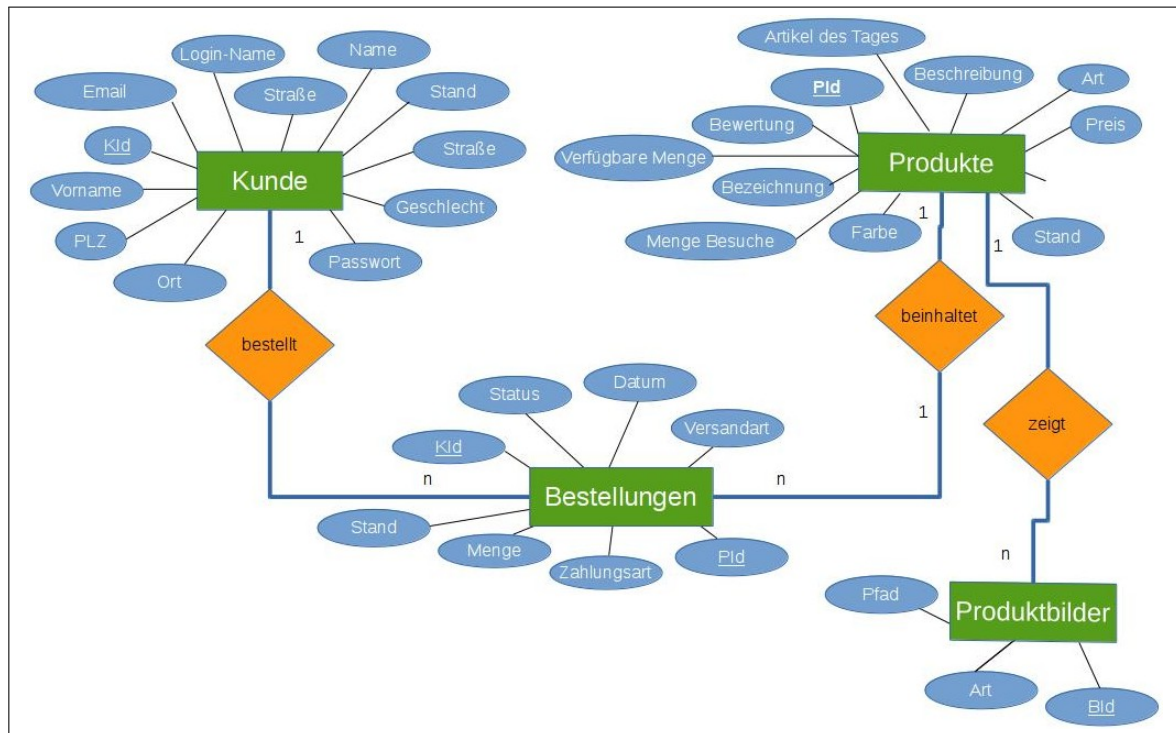


Abbildung 10: Erster Datenbankentwurf

Da es ursprünglich bemängelt wurde, dass das Projekt noch nicht den für das Modul benötigten Zeitaufwand aufweisen wurde, wurde beschlossen das Arbeitspaket „Bewertung“ mit einer Kommentarfunktion zu erweitern. Somit sollte der Kunde auch ein Kommentar für seine Bewertung äußern können. Dafür musste die Datenbank etwas angepasst werden. Vorher war einfach ein Attribut „Bewertung“ in der Relation „Produkte“ vorzufinden. Um die Kommentare für die Bewertung speichern zu können wurde die Datenbank um eine Relation „Kommentare“ erweitert. Dieses Relation beinhaltet ein Datenfeld, wo die Bewertung des Kunden (von 1-5 Sterne) und eins wo Bewertungstext gespeichert wird.

Als mit der Suchfunktion und der Navigation begonnen wurde fiel schnell auf, dass benötigte Attribute und Tabellen für die Navigation vergessen wurden. Somit wurde eine neue Relation „Produktkategorie“ zur Datenbank hinzugefügt. Dieses Relation steht mit der Relation „Produkte“ in Beziehung und beinhaltet alle Produktkategorien, die der Webshop anbieten soll. Unter Produktkategorien wird beim Webshop die Festlegung verstanden, ob das Produkt ein Ersatzteil, ein Fahrrad, Zubehör usw. ist. Eine zweite Relation, die mit der Relation Produkte in Beziehung steht, ist die Relation „Bauart“. Dieses Relation wird nur verwendet, wenn das Produkt ein Fahrrad ist. In dieser Relation sind alle beim Webshop angebotenen Bauarten aufgelistet. Unter Bauart wird bei diesen Webshop verstanden, ob es sich um ein Mountainbike, Trekkingbike usw. handelt.

Die *Abbildung 11* zeigt die endgültige Struktur der Schnittstellendatenbank mit allen nachträglich hinzugefügten Relationen.

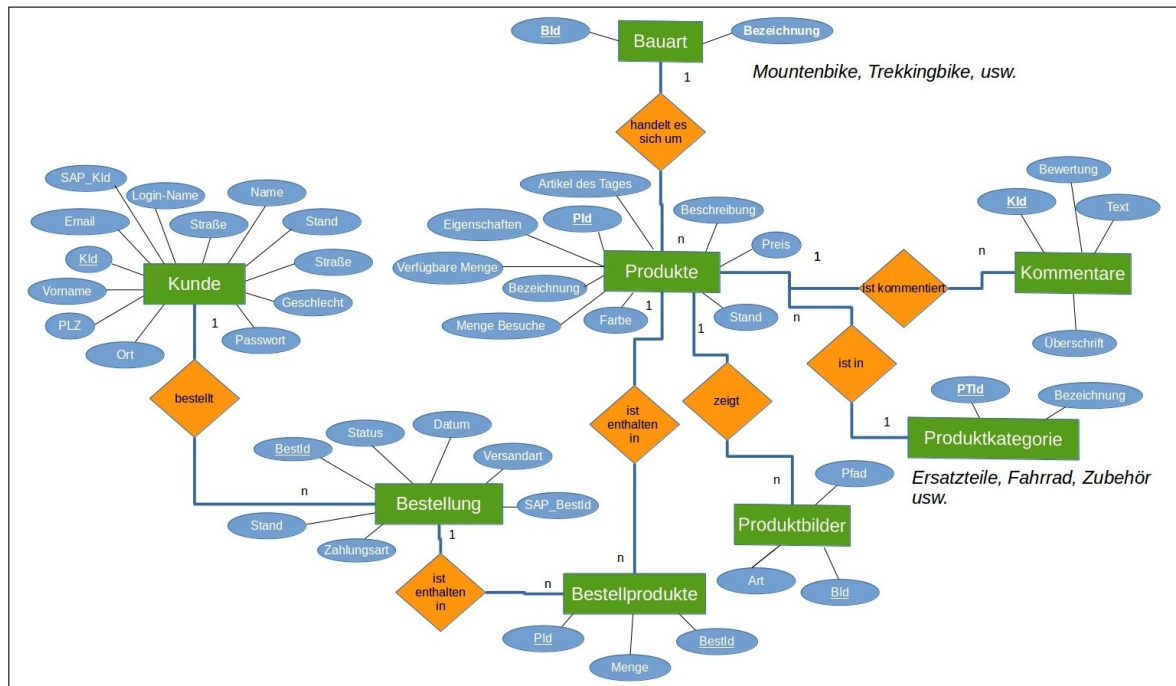


Abbildung 11: Endgültiger Datenbankentwurf

Wahl der PHP-API zum Datenbankzugriff

Um von PHP aus auf eine MySQL-Datenbank zugreifen zu können gibt es mehrere Möglichkeiten. In dieser Ausarbeitung werden die Zugriffsmöglichkeiten kurz verglichen und dann erläutert welche davon für das Projekt benutzt wurde.

Die erste Möglichkeit ist die PHP-API „ext/mysql“. Diese API ist veraltet und wird nicht mehr weiterentwickelt. Sie unterstützt noch nicht alle MySQL 5.1+ Funktionalitäten und ist stellenweise mit neuen PHP-Versionen nicht mehr nutzbar. Diese Erweiterung nutzt den prozeduralen Ansatz. Die nächste Möglichkeit ist die PHP-API „PDO“. Diese API unterstützt fast alle Funktionalitäten von MySQL 5.1+. Sie bietet zudem den Vorteil, dass die API auch zum Zugriff auf andere SQL-basierten Datenbanken genutzt werden kann. Diese API hat einen objektorientierten Aufbau. Die letzte Zugriffsmöglichkeit bietet die API „ext/mysql“. Sie unterstützt alle MySQL 5.1+ Funktionalitäten und kann objektorientiert und prozedural programmiert werden.

Die Auswahl fiel auf die API „ext/mysql“. Die Gründe hierfür waren, dass die API alle Funktionalitäten der neusten MySQL-Version unterstützt, regelmäßig auf die neuesten MySQL-Funktionen angepasst wird und schon Quellcode von alten Projekten übernommen werden konnte. Die *Abbildung 12* zeigt die Eigenschaften der drei APIs nochmal in einer Tabelle zusammen gefasst.

	ext/mysql	PDO_MySQL	ext/mysql
PHP version introduced	5.0	5.1	2.0
Included with PHP 5.x	Yes	Yes	Yes
Development status	Active	Active	Maintenance only
Lifecycle	Active	Active	Deprecated
Recommended for new projects	Yes	Yes	No
OOP Interface	Yes	Yes	No
Procedural Interface	Yes	No	Yes
API supports non-blocking, asynchronous queries with mysqlnd	Yes	No	No
Persistent Connections	Yes	Yes	Yes
API supports Charsets	Yes	Yes	Yes
API supports server-side Prepared Statements	Yes	Yes	No
API supports client-side Prepared Statements	No	Yes	No
API supports Stored Procedures	Yes	Yes	No
API supports Multiple Statements	Yes	Most	No
API supports Transactions	Yes	Yes	No
Transactions can be controlled with SQL	Yes	Yes	Yes
Supports all MySQL 5.1+ functionality	Yes	Most	No



Wahl

Abbildung 12: Überblick der PHP-APIs zum Datenbankzugriff

Die Konfigurationsdatei

Um auf eine MySQL-Datenbank zugreifen zu können werden üblicherweise mehrere Zugangsdaten benötigt. Die Zugangsdaten, die mindestens benötigt werden sind der Datenbankbenutzername, das Passwort zum Benutzernamen, der Datenbankname und die IP-Adresse des Servers. Damit diese Zugangsdaten nicht bei jedem Datenbankzugriff im Quelltext angegeben werden müssen, wurden diese Angaben in eine Konfigurationsdatei ausgelagert. Wird die Webseite später mal auf einen anderen Webserver betrieben, so müssen die Datenbankzugangsdaten nur in dieser Datei geändert werden. Die Konfigurationsdatei ist extra gesichert worden, damit sie nicht direkt im Browser geöffnet werden kann. Der Datei wurde als Dateiendung „*.php.inc“ angehängt. Mit einer bestimmten Zeile in einer Konfigurationsdatei für den Webserver wurde das Öffnen von „*.inc“-Dateien dem Browser untersagt. Die Konfigurationsdatei trägt den Namen „.htaccess“ und befindet sich im Home-Verzeichnis der Webseite. Mehr zur „.htaccess“-Datei kann im Abschnitt „Webshopsicherheit“ gelesen werden.

Der *Quellcode 1* zeigt die Konfigurationsdatei des Webshops. In dieser Datei sind die benötigten Datenbankzugangsdaten und die festgelegte Email-Adresse zu lesen, von der alle Bestätigungs-E-mails versendet werden sollen. Zudem sind in der Konfigurationsdatei die Bankdaten definiert, die den Kunden mitgeteilt werden, wenn er als Zahlungsart „Vorkasse“ ausgewählt hat.

```
1 <?php
2     //Angaben zur Verbindung mit der Datenbank
3     $benutzername = 'root';
4     $password = 'Passwort';
5     $datenbank = 'webshop';
6     $server = '127.0.0.1';
7
8     //Festlegung der Webshop-Emailadresse
9     $webshop_email_adresse = "webshop@fahrrad-gbi.de";
10    $webshop_email_name = "Webshop Fahrrad GBI";
11
12    //Konto-Daten der GBI für die Bestätigungs-Email
13    //(wenn Vorkasse von Kunden gewählt)
14    $bankdaten_gbi = array(
15        "inhaber" => "Global Bike Inc.",
16        "iban" => "DE08700901xxxxDemo",
17        "bic" => "1234567890B");
18    ?>
```

Quellcode 1: Die Konfigurationsdatei

Die Datenbankzugriffsklasse

Beim Webshop wird nicht direkt über die „MySQLi“-API auf die MySQL-Datenbank zugegriffen. Es wurde für den Datenbankzugriff extra eine Klasse entwickelt, die als Schnittstelle zwischen den Webshop und der „MySQLi“-API fundiert. Diese Klasse bestand schon von alten Projekten und wurde aus folgenden Grund entwickelt. Sie entstand zu den Zeitpunkt, als die PHP-Entwickler auf der PHP eigenen Webseite bekanntgaben, dass das Standard-MYSQL-API nicht mehr mit den nächsten PHP-Versionen unterstützt werden soll. Somit wurde entschlossen eine Klasse zu entwickeln, dass bei ähnlicher Situation der Nachfolger-API einfach nur der Quelltext in der Klasse geändert werden muss und nicht im ganzen Projekt überall bei gleicher Situation die API ersetzt werden muss. Zudem bietet die Klasse auch einen Vorteil, wenn zu einen späteren Zeitpunkt mal ein anderes Datenbanksystem verwendet werden soll. Bei dieser Situation müsste wie beim vorherigen Beispiel auch nur die Klasse verändert werden und nicht das ganze Projekt.

3.3.2 Arbeitspaket 6: Registrierungs- und Anmeldungsfenster designen und programmieren

Dieser Abschnitt beschreibt, wie beim Webshop die Registrierung und Anmeldung umgesetzt wurde. Zudem wird die Umsetzung einer Funktion des Webshops beschrieben, wo der Kunde seine Profildaten ändern kann.

Der Registrierungsvorgang

Auf dieser Webseite läuft die Registrierung, wie in den folgenden Sätzen beschrieben ab. Zunächst tippt der Neukunde die von der Webseite verlangten persönlichen Informationen in ein Formular ein und bestätigt die Eingabe durch einen Klick auf den Bestätigungsbutton. Wenn beim Webbrowser des Neukunden JavaScript nicht deaktiviert ist, überprüft der Webbrowser mit einem JavaScript die Eingabe bevor sie zum Webserver weitergeleitet wird. Die Eingabe wird nur weitergeleitet, wenn dieses vom JavaScript aus als gültig anerkannt wurde. Bei ungültiger Eingabe wird auf der Webseite eine Fehlermeldung ausgegeben. Der JavaScript ist keine endgültige Eingabeüberprüfung, dieser dient nur dafür unnötigen Datenverkehr zu minimieren. JavaScript besitzt den Nachteil, dass dieser beim Webbrowser deaktiviert werden kann. Bei deaktivierten JavaScript wird somit die Eingabe unüberprüft an den Webserver weitergeleitet. Somit muss die Eingabe nochmal beim Webserver auf Gültigkeit überprüft werden. Der Neukunde wird also erst in die Kunden-Tabelle der Datenbank geschrieben, wenn die serverseitig programmierten Überprüffunktionen die Eingabe als gültig angesehen haben. Wurde die Eingabe als ungültig anerkannt, so wird auch bei der serverseitigen Programmierung eine Fehlermeldung auf der Webseite angezeigt. Nachdem der Kunde erfolgreich in der Kundendatenbank angelegt wurde, bekommt der Neukunde eine Bestätigungsmail zu gesendet. Der Neukunde kann sich erst erfolgreich am System anmelden, wenn dieser den in der Email enthaltenden Link ausgeführt hat.

Das Passwort des Kunden wird nicht in Klartext in die Datenbank gespeichert. Bevor es in die Datenbank geschrieben wird, wird das Passwort mit einem „MD5-Hash-Algorithmus“ unkenntlich gemacht. Der MD5-Hashwert ist nicht mit einem Schlüssel wieder invertierbar. Er kann zur Speicherung von Passwörtern verwendet werden, da beim „MD5-Hash-Algorithmus“ der gleiche Input immer den gleichen Output erzeugen. Somit wird beim Login-Check einfach auch das eingegebene Passwort unkenntlich gemacht und verglichen, ob es mit dem Wert in der Datenbank übereinstimmt.

Die *Abbildung 13* zeigt den Registrierungsvorgang nochmal als BPMN dargestellt.

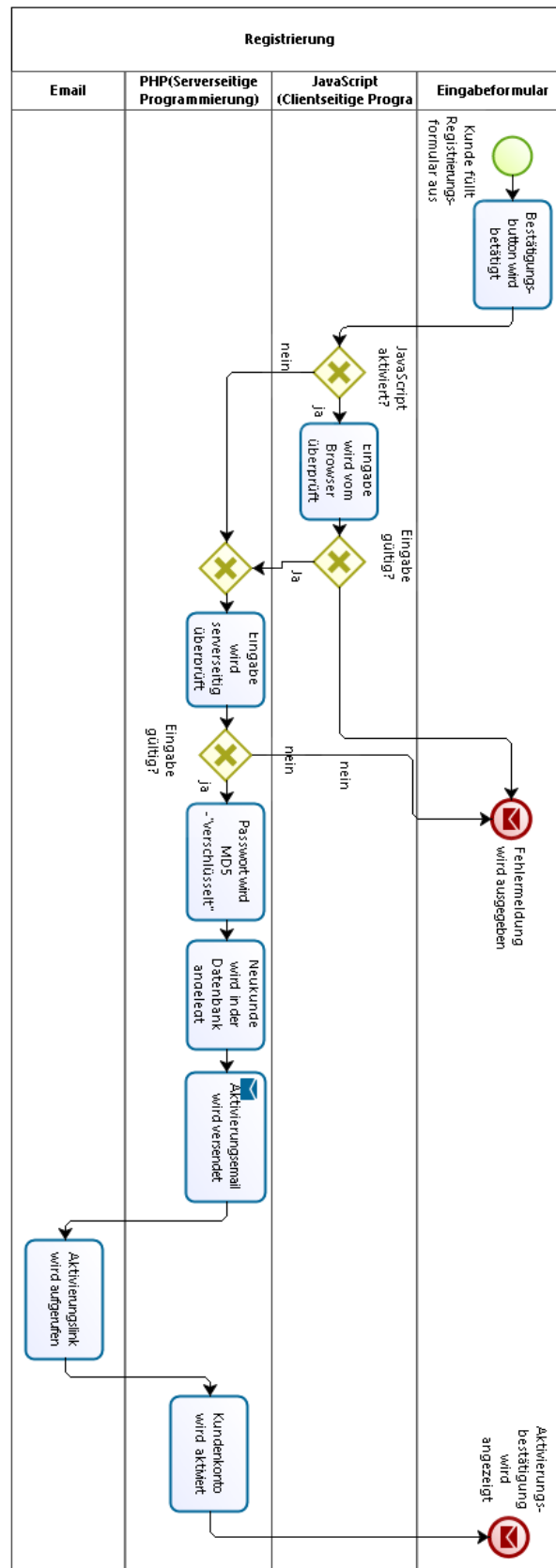


Abbildung 13: Der Registrierungsprozess

Der Anmeldungsvorgang

Der Anmeldungsvorgang läuft ähnlich wie der Registrierungsverfahren ab. Der Kunde tippt als Benutzernamen seine Email-Adresse und als Passwort das bei der Registrierung angegebene Passwort in das Anmeldeformular ein. Wie bei der Registrierung überprüft ein JavaScript die Eingabe, bevor diese an den Server weitergeleitet wird. Die vorherige Überprüfung ist wieder nur bei aktivierten JavaScript im Webbrowser möglich. Deswegen wird die Überprüfung bei der Anmeldung auch nochmal vom Server überprüft. Hat der Kunde eine gültige Eingabe getätigt, so wird die Email-Adresse und das Passwort mit den Datensätzen in der Kundentabelle der Datenbank verglichen. Um den Vergleich ausführen zu können wird bei der Anmeldung auch das Passwort durch den „MD5-Hash-Algorithmus“ unkenntlich gemacht. Wurde in der Tabelle ein Datensatz gefunden, wo Email-Adresse und Passwort übereinstimmen, so wird der Kunde eingeloggt. Wurde keine Übereinstimmung gefunden, so wird auf der Webseite eine Fehlermeldung ausgegeben.

Der Einloggvorgang läuft wie folgt umgesetzt. Die Anmelde-Email-Adresse wird in einer SESSION-Variable gespeichert. Der SESSION-Variable wurde den Namen „\$_SESSION[‘angemeldet’]“ gegeben. An Hand dieser Variable erkennt der Webshop, ob der Kunde angemeldet ist oder nicht. Wurde die Variable angelegt und enthält einen Wert, so wird dieses als angemeldet interpretiert. Die Abmeldung erfolgt einfach durch das Löschen dieser Variable. Eine SESSION-Variable behält die enthaltenden Daten über mehrere Seitenaufrufe. Wenn nicht mehr auf die Variable zugegriffen wird, werden die Daten dieser Variable automatisch nach einer bestimmten Zeit gelöscht. Ob der Kunde angemeldet ist muss in einer SESSION-Variable zwischengespeichert werden, da das HTTP-Protokoll verbindungslos ist. Es wird also jede HTTP-Anfrage isoliert betrachtet. Ohne die SESSION-Variable könnte sich somit der Webserver nicht merken, dass der Kunde angemeldet ist. Einer Session wird in PHP eine eindeutige „SESSION-ID“ zugeordnet. Diese SESSION-ID wird entweder als Cookie im Browser abgespeichert oder immer der Seiten-URL mit übergeben. Wird die SESSION-ID in einen Cookie im Browser abgespeichert, so wird diese beim Schließen des Browsers gelöscht. Die Daten, die in der SESSION-Variable abgespeichert sind werden auf der Serverseite zwischengespeichert und nicht im Browser. Die SESSION-ID wird von Browser bei jeder Anfrage an den Server weitergeleitet. An Hand dieser SESSION-ID kann der Server die SESSION-Variable zu den entsprechenden Browser zuordnen.

Die clientseitige Überprüfung

Wenn JavaScript aktiviert ist, werden, wie oben schon erwähnt, die Formulareingaben zuvor im Browser überprüft. Dafür wird beim Formular den Attribut „onsubmit“ eine JavaScript-Funktion übergeben. Die Formulareingabe wird hierbei nur an den Server weitergeleitet, wenn diese Methode als Rückgabewert „true“ zurückgibt. Die JavaScript-Funktion überprüft bei der Eingabe, ob überhaupt ein Text in die Textfelder eingegeben wurde und die Eingabe gültig ist. Zum Beispiel darf eine Postleitzahl nur aus 5 Ziffern bestehen. Die clientseitige Programmierung findet bei der Anmeldung in der Datei „/Funktionen/JS/anmeldung.js“ und bei der Registrierung in der Datei „/Funktionen/JS/registrierung.js“ statt.

Der *Quellcode 2* zeigt in vereinfachter Version das Login-Formular. Beim „form“-Tag kann das „onsubmit“-Attribut gefunden werden. Dieses Attribut startet beim Klick auf den Submit-Button die enthaltende JavaScript-Funktion. Das Attribut lässt nur die Durchführung des Submits zu, wenn die JavaScript-Funktion als Rückgabewert den Wert „true“ zurückliefert.

Um unnötigen Quelltext zu umgehen, wurde ein Teil des Eingabeüberprüfungsquellcode in eine andere JavaScript-Datei ausgelagert. Der ausgelagerte Quellcode wird bei jedem Formular benötigt. Dieser Quellcode überprüft, ob alle Pflichtfelder ausgefüllt sind. Die ausgelagerte Funktion trägt den Namen „sindAlleFelderAusgefüllt“ und befindet sich in der Datei „/Funktionen/JS/vorcheck_std_funktionen.js“. Der Funktion wird beim Aufruf drei Parameter übergeben. Der erste Parameter erhält ein Array. Dieses Array enthält die Feldnamen der Textfelder, wovon die Eingabe überprüft werden soll. Der zweite Parameter enthält wiederum ein Array. Dieses Array enthält die Meldungswörter, die in der Fehlermeldung für das jeweilige Textfeld eingesetzt werden sollen. Die Meldungswörter werden in die Fehlermeldung eingesetzt, wenn das Textfeld nicht gefüllt ist. Beim letzten Parameter wird der Name des Formulars angegeben, worauf sich die Textfelder befinden. Sind nicht alle Felder ausgefüllt, erstellt die Funktion eine Fehlermeldung und gibt diese als Rückgabewert aus.

In die Auslagerungsdatei wurden zudem Funktionen zur Überprüfung der Email-Adresse oder ob die Eingabe eine Zahl ist abgelegt. Die Funktionen in den Dateien „/Funktionen/JS/anmeldung.js“ und „/Funktionen/JS/registrierung.js“ rufen die ausgegliederten Funktionen auf und führen noch eine genauere Überprüfung durch. Die Funktionen testen zum Beispiel, ob das Passwort mit der Passwortwiederholung übereinstimmt.

```
1  <!-- JavaScript zur Überprüfung der Eingabe wird zur Webseite importiert -->
2  <script type="text/javascript"
3      src="./Funktionen/JS/vorcheck_std_funktionen.js" >
4  </script>
5  <script type="text/javascript" src="./Funktionen/JS/anmeldung.js" >
6  </script>
7
8  <form method="post" action="..." name="form_login" id="login"
9      onsubmit="return vorcheckEingabe_anmeldung('form_login');">
10
11     <!-- Textfeld zur Eingabe der Login-Email-Adresse -->
12     <label for="anmedlung_email">Email:</label>
13     <input type="text" name="anmedlung_email" maxlength="50"
14         class="lang_reg" />
15
16     <!-- Textfeld zur Eingabe des Passworts -->
17     <label for="anmedlung_passwort">Passwort:</label>
18     <input type="password" name="anmedlung_passwort" maxlength="50"
19         class="lang_reg" />
20
21
22     <!-- Fehlermeldungen bei ungültiger Eingabe -->
23     <p class="text_fehlermeldung" id="fehlermeldung_login">
24         <?php /* Fehlermeldungstext */ ?>
25     </p>
26
27     <!-- Der Submit-Button -->
28     <input type="submit" value="Login" title="Login"/>
29 </form>
```

Quellcode 2: Login-Formular (vereinfacht)

Serverseitige Programmierung

Wie bei der clientseitigen Programmierung wurde auch bei der serverseitigen Programmierung die Methode zur Überprüfung, ob alle Pflichtfelder ausgefüllt sind ausgelagert. Bei der serverseitigen Programmierung wurde hierfür die Technik der Vererbung genutzt. Serverseitig wurde der Webshop objektorientiert programmiert. Für die zuvor genannte Methode wurde die Klasse „EingabeCheckGrundlegend“ erstellt. Diese Klasse wurde an alle anderen Klassen vererbt, die Formulareingaben entgegen nehmen müssen. Für die Registrierung, Anmeldung und Änderung des Kundenprofils ist die Klasse „Kunde“ entwickelt worden. Sie enthält Methoden, mit der die Formulareingabe geprüft, ein neuer Kunde angelegt, ein Kunde verwaltet und die Anmeldung durchgeführt werden kann.

Instanziiert wird die Klasse durch Skripte des Ordners „/Funktionen/PHP“. Registriert sich ein neuer Kunde am Webshop so wird von den Registrierungsformular der Skript „registrierung_durchfuehren.php.inc“ des zuvor genannten Ordners aufgerufen. Dieser Skript instanziiert die Klasse „Kunde“, überprüft die Formulareingabe und legt bei gültiger Eingabe den Kunden an. Zur Überprüfung der Eingabe und zum Anlegen des Kunden ruft der Skript die

benötigten Methoden der Klasse „Kunde“ auf.

Weitere Skripte, die mit der Klasse „Kunde“ arbeiten, sind zum Beispiel die Skripte „anmeldung_durchfuehren.php.inc“, „kunde_aktivieren.php.inc“ und „kunde_aendern.php.inc“. Der erste der Skript der Liste ist für die Anmeldung des Kunden zuständig, der zweite aktiviert den Kunden, wenn der Aktivierungslink der Aktivierungsemail angeklickt wurde und der letzte Skript ist für das Ändern der Anschrift oder Benutzerkennung des Kunden zuständig. Die Klasse „Kunde“ legt alle Daten in der Datenbank ab. Wird z.B. die Instanzmethode „get_vorname_from_kid(\$kid)“ aufgerufen, so fragt die Methode über einen Datenbank-Select den Vornamen des Kunden von der Datenbank ab. Das Java-Schnittstellenprogramm der SAP-Gruppe lauscht die ganze Zeit, ob sich etwas an der Datenbank ändert. Kam es zu einer Änderung in der Datenbank, so wird die Änderung im SAP-System eingetragen.

Der Loginbereich im Header

Der Loginbereich im Header wurde mit einen sogenannten „Flyout“ umgesetzt (siehe *Abbildung 14*). Ein Flyout beblendet ein Fenster auf der Webseite ein, wenn über ein bestimmtes Symbol mit der Maus gefahren wird. Der Loginbereich bei der Webseite wird also erst angezeigt, wenn sich die Maus auf den Anmeldesymbol befindet.

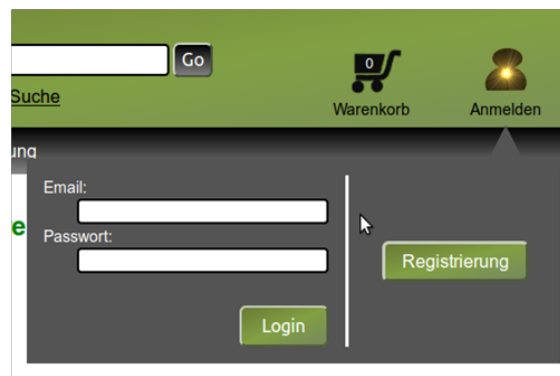


Abbildung 14: Flyout: Loginbereich

Die Besonderheit bei den Flyout des Webshops ist, dass es auch bei deaktivierten JavaScript funktioniert. Das Flyout wurde mit den in CSS 2.0 eingeführten „Kindselektor“ umgesetzt. Mit diesen „Kindselektor“ ist es eingeschränkt möglich mit CSS eventbasiert zu programmieren. Um diese CSS-Funktion nutzen zu können muss aber folgende Bedingung erfüllt sein: Das Fenster, dass angezeigt werden soll muss ein Kindelement des Elements sein, welches das Ereignis auslöst. Die Syntax das „Kindselektor“ sieht wie im *Quellcode 3* zeigt aus. Am Anfang steht die Bedingung, die am Elternelement erfüllt sein muss. Beim Webshop wäre es die Bedingung, dass die Maus sich auf den Anmeldesymbol befinden soll. Diese Bedingung sieht in CSS geschrieben folgendermaßen aus: „accountCell: hover“. Darauf folgt ein „>“-Zeichen. Dieses Zeichen legt fest, dass bei erfüllter Bedingung nicht die Style-Anweisung des Elternelements verändert werden soll, sondern eines Kindelementes. Nachfolgend wird das Kindelement angegeben, wo die Style-Anweisung geändert werden soll. Der letzte Teil der Style-Anweisung legt das neue Aussehen des Kindelements fest. Beim Webshop wird hier

festgelegt, dass das Kindelement sichtbar werden soll. Ein Paar Zeilen vorher definiert eine Anweisung, dass das Kindelement unsichtbar sein soll. Bei gültiger Bedingung wird diese Anweisung einfach von der eben genannten Anweisung überschrieben.

```
1  /* Blendet den Loginbereich standardmäßig aus */
2  .popupRegistrierung, .popupWarenkorb{
3      ...
4      visibility:hidden;
5      ...
6  }
7  /*
8   * Zeigt den Den Loginbereich an,
9   * wenn die Maus sich auf auf dem Anmeldesymbol befindet
10  */
11  .accountCell:HOVER > .popupRegistrierung, .popupRegistrierung:hover{
12      visibility:visible;
13  }
```

Quellcode 3: Loginbereich: Umsetzung des Flyouts mit CSS 2.0

Kundenprofil ändern

Hat sich eine Kunde angemeldet, so bietet der Webshop ihn die Möglichkeit die bestehenden Login-Daten und die persönlich Anschrift zu ändern. Erreicht können die Formulare zur Änderung der Daten über ein Flyout. Wenn der Kunde angemeldet ist wird das Flyout unter den Anmeldesymbol zu einen Menü geändert (siehe *Abbildung 15*).

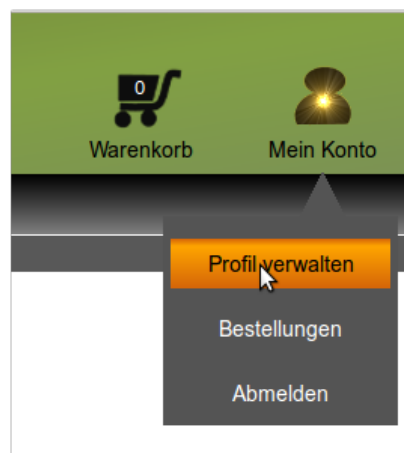


Abbildung 15: Flyout: Kunde angemeldet

Bei diesem Menü muss der Eintrag „Profil verwalten“ gewählt werden, um die bestehenden Login-Daten und die persönlich Anschrift ändern zu können.