Gliederung

1.Ausbildungsbetrieb und Einsatzgebiet....2

2.Eingesetzte Entwicklungsumgebung....2
2.1.Hardware....2
2.2.Software....3
2.2.1.lokal benutzte Software....3
2.2.2.remote benutzte Software....3

3.Projekte....5
3.1.Entwicklung von Intranettools....5
3.1.1.Bannerverwaltung....6
3.1.2.Produktübersicht....7
3.2.Entwicklung von Internetseiten....8
3.2.1.Kunden-werben-Kunden....8
3.2.2.Bestellprozess....9
3.2.3.Engine-Version der Website....13

4.Weitere Aufgaben....17
4.1.Auswahl und Erstellung von HTML-Templates....18
4.2.Internet-Recherche über neue Produkte....18
4.3.Erstellen von Grafiken....18

5.Persönliche Meinung....18

6.Abkürzungsverzeichnis....19

7.Literaturverzeichnis....20
7.1.Print-Medien....20
7.2.Online-Medien....21

+++ AKTUALISIEREN +++

1. Ausbildungsbetrieb und Einsatzgebiet

Das erste Praktikumssemester leistete ich bei der Firma WWW-Service Online-Dienstleistungen GmbH in Neutraubling. Das Unternehmen ging 1995 aus einer Abteilung des ABC-Bücherdienstes, der heute unter dem Namen Amazon firmiert, hervor und bietet ein komplettes Webhosting und Webhousing - Angebot. Im Oktober 1998 schloss sich das Unternehmen mit der amerikanischen Verio, Inc. zusammen. Schließlich wurde Verio während meiner Zeit im Unternehmen von der Nippon Telephone & Telegraph Public Corporation (NTT[1] Communications) aufgekauft.

Der WWW-Service gliedert sich in folgende Abteilungen:

- Buchführung
- Sales
- Domainregistrierung
- Technischer Support
- Technische Dokumentation
- Interne Programmierung
- Interne Technik
- Webdesign
- Marketing und PR

Zusätzlich sitzt in Neutraubling auch die Tochterfirma RapidSite, die sich dem Reseller-Geschäft widmet. Eingesetzt wurde ich in der Abteilung Webdesign, die sich in erster Linie um die Präsentation beider Firmen im Internet und der Programmierung von Internet- und Intranet-Tools mit Datenbank-Anbindung kümmert.

2. Eingesetzte Entwicklungsumgebung

2.1. Hardware

Die Software-Entwicklung führte ich auf einem Intel Pentium II 300 mit 64 MB Arbeitsspeicher, der im Laufe des Praktikums auf 128 MB RAM erweitert wurde, durch. Als Staging- und Veröffentlichungs-Server dienten Silicon

Graphics Origin 200 QC mit je zwei 64 Bit R10.000 MIPS Prozessoren und 640 MB RAM. Gelegentlich wurden Dedicated Server von Sun benutzt.

2.2.Software

2.2.1.lokal benutzte Software

Zum Einsatz auf dem Entwicklungsrechner kamen hauptsächlich:

- Microsoft Windows NT 4.0 (Service Pack 6),
- Microsoft Office 2000 Professional (vor allem Outlook, auch Access, Excel und Word),
- Microsoft Visio 4.0 und 5.0
- Adobe Photoshop 5.0 und 6.0,
- Adobe Illustrator 7.0,
- Macromedia Dreamweaver 3.0,
- PHPCoder R2 beta mit PHP 4.0.3pl1 und
- UltraEdit 7.0.

2.2.2.remote benutzte Software

Auf den Servern wurde mit folgender Software gearbeitet:

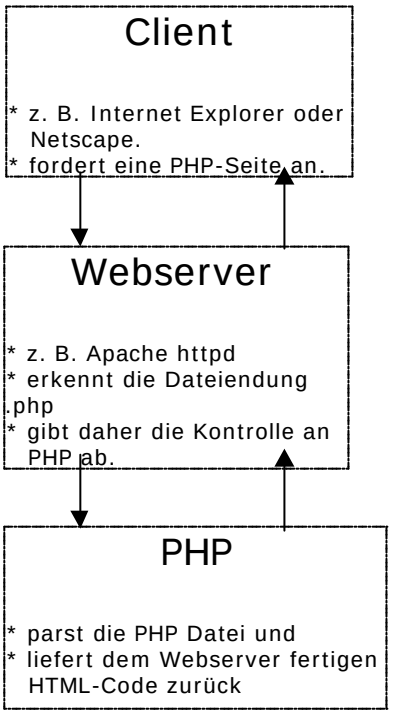
- Sillicon Graphics Irix64 nexus 6.4 auf den Shared Hosting Accounts,
- Sun Solaris 5.6 auf den Dedicated Servern,
- Apache httpd 1.3.4,
- PHP 3.0.9 als CGI-Modul und
- MySQL 3.22.23b.

Ein Grossteil meiner Arbeit bestand aus der Erstellung von Skripten mit Datenbankanbindung. Zum Einsatz kam die Open-Source Programmiersprache PHP[2] (PHP Hypertext Processor) in der Version 3.0.9. Sie zeichnet sich durch C-ähnliche Syntax und eine Vielzahl an Funktionen -auch für die Steuerung von verschiedenen Datenbanken- aus.

PHP ist eine serverseitige Programmiersprache. Die Ausführung der

Programme läuft daher wie folgt ab (nach [2a]):

- Der Anwender fordert über das HTTP[3]- oder HTTPS[4]-Protokoll eine PHP Seite an.
- Der Webserver übergibt die gewünschte Seite an PHP.
- PHP interpretiert den Code und liefert den HTML Text an den Webserver zurück.
- Dieser liefert die Seiten an den Client des Anwenders.



Der Browser des Anwenders erhält also PHP-generierten HTML-Code. PHP Anweisungen sind nicht mehr enthalten. Im Gegensatz dazu steht die clientseitige Programmierung beispielsweise mit JavaScript, die ich im Rahmen meines Praktikums ebenfalls streifte. Hier wird der Programmcode zum Client geschickt und erst dort ausgeführt. Die Problematik bei clientseitigen Techniken ist die Abhängigkeit von der Konfiguration des Clients. Wegen diverser Sicherheitsmängel wird JavaScript oft deaktiviert. Auch sichere JavaScript Anwendungen (wie z. B. die weit verbreiteten Rollover Effekte auf Navigationselementen) werden nicht mehr angezeigt.

Aus diesem Grund wird bei Verio versucht, die Seiten möglichst serverseitig anzulegen. Auf JavaScript wird wo es möglich ist verzichtet. So war auch

eine meiner ersten Aufgaben die Seiten von bereits vorhandenen Formularüberprüfungen mit JavaScript zu befreien und entsprechende Überprüfungen in PHP zu implementieren.

Um Webapplikationen mit Datenbankanbindung zu realisieren wurde wurde das ebenfalls frei verfügbare mySQL 3.22.23b gewählt. mySQL ist eine weit verbreitete, vor allem für kleine und mittlere Projekte geeignete Datenbank. mySQL arbeitet als RDBMS[5] (Relational Data Base Management System), d. h. es besteht aus miteinander verbundenen Tabellen, die Datenwerte enthalten. Alle Datensätze sind mit Hilfe von Primärschlüsseln eindeutig identifizierbar. Über SQL[6], die Structured Query Language können alle relationalen Grundoperationen und mengenorientierte Änderungen an den Daten vorgenommen werden.[7]

3.Projekte

Neben der Erstellung von Internetpräsentationen für Kunden und anfallenden Gestaltungs- und Pflegearbeiten an den Seiten von Verio wurden folgende Projekte von mir ausgeführt:

3.1.Entwicklung von Intranettools

Die ersten zwei Projekte, die hier beleuchtet werden sollen, dienen der internen Informationsverwaltung im Betrieb.

3.1.1.Bannerverwaltung

Wie alle grossen Firmen wirbt auch Verio durch die Schaltung von Werbebannern auf frequentierten Internetseiten. Eine EDV[8]-gestützte Verwaltung der produzierten Werbebanner war dringend erforderlich. Vorgabe war der Einsatz als Informations-Tool im Intranet das sämtliche Banner verwaltet, bereits eingesetzte kennzeichnet und die Möglichkeit bietet, die sog. Clickrate und die Anzahl der Einblendungen zu erfassen. Diese Daten werden vom Adserver-Anbieter, der sich um die Platzierung der Werbung kümmert, zurückgeliefert. Auch eine Auswahl der Werbebilder anhand der Bildgröße, der Dateigröße, der Anzahl der Animationsebenen (bei animierten GIFs[9]) und des Autors sollte möglich sein.

In der Projektvorbereitung einigte man sich auf die Realisation mit PHP und mySQL. Die Seiten wurden auf einem öffentlich zugänglichen HTTP-Server abgelegt, um der Medienagentur einen Zugriff auf die Daten zu ermöglichen. Der erforderliche Zugriffsschutz wurde durch die lokale Steuerungsdatei ".htaccess" des Apache Webservers bewerkstelligt. mySQL bietet die Möglichkeit Binärdaten (wie in diesem Fall die Bilder im JPEG[10]- und GIF-Format) als Datentyp "Blob[11] (Binary Large Object)" zu speichern. Wir untersuchten diese Option, rückten jedoch wieder davon ab, da in der PHP Mailingliste[12] von sehr schlechter Performanz dieser Lösung berichtet wurde. Also wurden die URLs[13] der Bilddateien in der Datenbank abgelegt. Um Abfrage-und Eingabe-Frontend auch optisch in das Intranet zu integrieren wurde ein externes Stylesheet (CSS[14]) eingebunden. (Eine grosse Hilfe beim Erlernen der CSS-Syntax war SelfHTML von Stefan Münz[14a]). Das Benutzerinterface gliedert sich in zwei Frames: im oberen werden die Abfragekriterien festgelegt, im unteren präsentieren sich dann tabellarisch die Ergebnisse. Diese werden mit einer SQL-Abfrage wie dieser aus der Datenbank geholt:

```
SELECT * FROM banner WHERE banner.ID != 0 AND
banner.Bildgroesse="234x60" AND banner.Ebenen <= "1"
AND banner.erstellt_von = "gable" ORDER BY banner.ID
ASC
```

Nach der Eingabe vieler bereits vorhandener Banner samt Erfolgsdaten konnte das Tool der Marketing-Abteilung vorgestellt und nach einigen kleinen Korrekturen übergeben werden.

3.1.2.Produktübersicht

Diese Applikation sollte die Arbeit für die Abteilung Sales erleichtern. Ziel war es eine vollständige Übersicht der angebotenen Produkte, samt Preisen und Leistungsmerkmalen zu generieren. Eine einfache Zusammenstellung als HTML[15]-Datei schied wegen der Fülle an Daten aus. Vielmehr sollte es möglich sein, die Abfragekriterien von Fall zu Fall auszuwählen.

Es zeigte sich, dass hier eine hervorragende Bedienbarkeit und schnelle

Ablesbarkeit der Daten wichtig ist. Entsprechende Informationsdesign wurde durch den Einsatz von sogenannten Presets unterstützt: die zahlreichen Leistungsmerkmale wurden in Gruppen aufgeteilt.

- „Preise“: Angaben zu den zu entrichtenden Gebühren
- „Basics“: Angaben zu Speicherplatz, Transfervolumen und Mail-Accounts
- “Details”: Angaben zur Serverplattform, installierten Erweiterungen, usw. und
- „Komplett“: alle Angaben

Nach der Auswahl der Presets sollte dennoch eine weitere Einschränkung der Abfrage möglich sein. Auch diese Anwendung würde im Intranet zur Verfügung gestellt werden.

Wieder entschieden wir uns für die Umsetzung in PHP und mySQL. Zunächst wurde eine Datenbank mit den kompletten Produktdaten erstellt. Als Eingabemaske diente der Einfachheit halber das frei verfügbare phpmyAdmin. Das Ausgabe-Frontend wurde mittels PHP erzeugt. Um ein einheitliches Konzept bei den Abfragewerkzeugen im Intranet zu gewährleisten, entschied ich mich wieder für die Darstellung von zwei Frames: im oberen werden die Eingaben getätigt, unten können die Ergebnisse in Tabellenform abgelesen werden. Im Gegensatz zur Bannerverwaltung sind nun wesendlich mehr Informationen zu berücksichtigen. Die Abfrage erfolgt wieder über eine SQL-Anfrage, die mit dem PHP-Befehl `mysql_query()` abgesetzt wird. Die Ergebnisse der Abfrage werden tabellarisch aufbereitet und ausgegeben.

Nach einigen nachträglichen Erweiterungen (wie z. B. einer Exportfunktion in eine Textdatei und der Möglichkeit, die Tabellen auszudrucken) konnte die Arbeit an diesem Projekt erfolgreich abgeschlossen werden.

3.2.Entwicklung von Internetseiten

Als großes Ärgernis entpuppte sich schon kurz nach meinem Eintritt in die Firma der teilweise erhebliche Arbeitsaufwand, der bei der Wartung des eigenen Internetauftritts entstand. So mussten z. B. bei einer Preisänderung

sämtliche Seiten durchsucht und die Daten ausgebessert werden. Abhilfe versprach eine Dynamisierung der Seiten.

3.2.1.Kunden-werben-Kunden

Als erster Schritt sollte die Kunden-werben-Kunden Rubrik automatisiert werden. Für das Werben eines Neukunden können die Kunden aus einem Angebot an Sachprämien wählen. Diese sollen regelmässig im Rhythmus von sechs Wochen gegen neue ausgetauscht werden, um den Service attraktiv zu halten. Angedacht war eine Vereinfachung des Arbeitsablaufs durch die einmalige Definition der Prämien samt Erklärungstext, Bildern und Zeitspanne, in der sie angezeigt werden sollten. Dem Anwender soll neben diesen Informationen auch ein Link auf eine vergrösserte Darstellung der Prämie präsentiert werden.

Ich entschied mich für eine Verwaltung der Daten in einer Datei mit PHP Variablendefinitionen, da eine Datenbankanbindung hier überdimensioniert gewesen wäre und die Pflege der Prämiendaten höchstens alle 3-4 Monate anfallen wird.

Die Variablendefinitionen werden in die eigentliche Anwendung mit der `include()`-Funktion eingebunden. Das Programm zählt in einer Schleife sämliche Prämien durch, prüft sie auf Aktualität und zeigt sie gegebenenfalls an. Dabei wird ein Link zu einem vergrösserten Zoombild generiert. Die Prüfung wurde durch einen Vergleich von Timestamps realisiert. Ein Timestamp ist die Anzahl von Sekunden, die seit dem 1. Januar 1970 auf dem Server vergangen ist. PHP bietet Funktionen, die die Umrechnung eines Datums in ein Timestamp und zurück ermöglichen.

Das Ergebnis ist auf www.www-service.de/website/kontakt/kundenwerben.php3 zu sehen. Eine Anleitung zum Warten der Variablendefinitionen wurde der Webdesign-Rubrik des Intranets hinzugefügt.

3.2.2.Bestellprozess

Wichtigstes Element der Website ist der Bestellprozess, über den die

überwiegende Mehrheit des Umsatz der Firma erzielt wird. Da der alte Prozess nicht mehr den Anforderungen gewachsen und sehr schlecht wartbar war, wurde die Planung eines neuen Bestellvorgangs in Angriff genommen. Der Anforderungskatalog sah folgene Eigenschaften vor:

- Datenintegrität:
  Die Produktdaten sollten aus einer zentralen Datenbank ausgelesen werden, um den Wartungsaufwand im Falle einer Änderung möglichst klein zu halten.
- Hohe Performanz und Ausfallsicherheit:
  Die Integration der Produktdaten per Datenbankanbindung darf den Seitenaufbau nicht merklich verlangsamen. Auch bei einem Ausfall der Datenbank muss das System funktionieren.
- Ausschluss von fehlerhaften Eingaben:
  Der Anwender muss möglichst selbsterklärend durch den Prozess geführt werden. Zusätzlich soll ein Hilfesystem zur Verfügung stehen, dass im richtigen Moment umfassende Information zum aktuell auszufüllenden Feld zur Verfügung stellt. Gibt der Anwender sinnlose Daten ein, soll er darauf hingewiesen werden.
- Integration sämtlicher Bestellvorgänge unter einer Engine:
  Bisher wurden verschiedene Formulare für die Bestellung von Shared Hosting Accounts, Virtual Privat- und Dedicated Servern benutzt. Eine zusätzliche Bestellmöglichkeit für Domain-only Produkte sollte ebenfalls hinzugefügt werden.
- Modularisierung:
  Ein Hinzufügen von neuen Produkten soll problemlos möglich sein. Auch eine Änderung des Exportweges (Mail, PGP[16]-Mail, Datenbankeintrag, ...) muss leicht nachrüstbar sein.

Diese Vorgaben führten zu folgenden Überlegungen: Die Datenintegrität kann durch einen Zugriff auf die in Kapitel 3.1.2. erwähnte Produktdatenbank gewährleistet werden. Um dennoch den Aufbau der Seiten nicht zu verlangsamen wurde der Umweg über eine CSV[17]-Datei gewählt. Diese soll regelmässig aus der Datenbank erstellt werden.

Für die Modularisierung sorgt eine Aufteilung des Prozesses in

verschiedene Dateien; für jeden logischen Schritt eine:

- Account und Registrant
- technischer Ansprechpartner
- Rechnungsadresse
- Bankverbindung oder andere Zahlungsart
- Zusatzoptionen
- Korrekturmöglichkeit und AGB-Bestätigung
- Dankesseite und eigentliche Bestelldurchführung

Im Falle einer Produktneueinführung muss so nur die Datenbank geändert werden, sofort steht das neue Produkt im Bestellvorgang zur Verfügung. Auf diesem Weg wurde auch die Integration sämlicher Produkte bewerkstelligt.

Um der Abteilung Domainregistrierung die Arbeit zu erleichtern werden die eingegebenen Daten mit regulären Ausdrücken verglichen. Bei einem Fehler wird der aktuelle Schritt, diesmal mit einer Fehlermeldung nochmals angezeigt. Andernfalls wird entsprechend der Programmlogik weiterverzweigt. So kann man z. B. auf der ersten Seite (Registrant) die Checkboxen „Registrant ist zugleich Ansprechpartner“ und „Registrant ist zugleich Rechnungsempfänger“ anklicken. In diesem Fall sollen die Schritte 2 und 3 natürlich nicht mehr angezeigt werden.

Das Hilfesystem besteht aus einem JavaScript, welches kleine Popup-Fenster erzeugt. Diese erscheinen, wenn man sich mit der Maus über einem Punkt neben der Eingabefläche steht. Bewegt man die Maus wieder, verschwindet es. Um diesen Effekt zu erzielen wurden Layer eingesetzt, eine Eigenschaft von HTML 4.0. Dies funktioniert zwar bei Anwendern mit älteren Browsern nicht, aber da das Hilfesystem kein „Core-Feature“ ist wurde dieser Nachteil in Kauf genommen. Der Aufruf des JavaScripts wird jeweils dynamisch von PHP erzeugt.

Sowohl die in der Hilfe als auch die restlichen Texte kommen aus einer PHP-Variablendefinitionsdatei und sind damit leicht auszutauschen. Denkbar ist somit auch eine spätere Lokalisierung des Prozesses, um ausländischen Kunden die Bestellung zu erleichtern.

Das Resultat ist auf www.verio.de/website/order zu begutachten. Mit der Einbindung in die eigentliche Seite wird noch gewartet, bis das Fakturierungsprogramm die zusätzlichen Daten wie den technischen Ansprechpartner aufnehmen kann. Um auch meinen Nachfolgern die Möglichkeit zu geben dieses Projekt fortzuführen legte ich Wert auf eine gute Dokumentation der Arbeit. Mit Visio wurde folgendes Flussdiagramm erstellt und umfangreiche Erklärungsseiten wurden im Intranet veröffentlicht.

+++ Order.eps einfügen +++

3.2.3.Engine-Version der Website

Die zuvor aufgeführten Projekte dienten quasi als Vorbereitung für eine Umstellung der Webseiten für den WWW-Service. Bisher waren die Seiten größtenteils in statischem HTML geschrieben, erstellt mit Microsoft FrontPage. Dies führte zu den schon geschilderten Nachteile im Bezug auf die Wartbarkeit. Auch kam es manchmal vor, dass die Marketing-Abteilung Inhalte auf der Website änderte und somit unser lokal gespiegelter Inhalt des Webservers inkonsistent wurde. Abhilfe hätte ein sogenanntes Content Management System, kurz CMS[18] bringen können. Die Hauptaufgaben eines CMS sind folgende:

- Trennung von Layout und Inhalt:
  Die Auszeichnung der HTML-Seiten, wie z. B. Schriftart und -farbe werden nicht mehr mit dem Inhalt gemischt in einer Datei gehalten, sondern getrennt verwaltet. Ein erster Ansatz hierzu ist auch ohne Programmierung möglich (CSS). Weitere Vereinheitlichung (auch unter dem Stichwort „Grundsatz der Monotonie“) ist jedoch erst mit CMS zu erreichen.
- Assetmanagement:
  Verwaltung der verwendeten Komponenten (Texten, Tabellen, Bildern, ...). Obwohl für Verio nicht unbedingt notwendig, könnte Assetmanagement mehr Überblick in die Vielzahl an Dateien auf dem Webserver bringen und helfen Dubletten zu vermeiden.
- Workflow-Organisation:
  Mit einem Redaktionssystem sollten sich die Aufgabenbereiche genau abgrenzen lassen. Grundlage dieses Prozesses ist eine Benutzer- und Rechteverwaltung. Bei Verio ist dieses Konzept auch nicht unbedingt erforderlich und kann in einer ersten Implementierung ausgespart werden.

Aufgrund der teilweise extrem hohen Kosten (bis zu 500000 Euro) oder des enormen Umstellungsaufwands kamen kommerzielle Systeme jedoch nicht in Frage. Auch die wenigen Systeme die unter der GPL[19] (GNU Public License) standen konnten nicht vollständig überzeugen. Man kam zu der Überlegung, dass es am besten sei, einige Vorüberlegungen anzustellen, diese den Vorgesetzten zu präsentieren und bei einem positiven Ergebnis ein solches

System gemäß den Anforderungen im Unternehmen selbst zu entwickeln.

Die Webseiten sind nach einem immer gleichen Muster aufgebaut:

- ein statischer Header mit Firmenlogo und einigen Navigations-elementen,
- ein von Seite zu Seite sich ändernder Navigationskasten auf der linken Seite,
- ein ebenfalls von der angezeigten Seite abhängigen Inhalt auf der rechten Seite und
- ein statischer Footer mit Textnavigation und Kontakt-möglichkeiten.

Es schien sinnvoll, zunächst die statischen und auf jeder Seite identischen Elemente als externe Dateien zu halten und zur Laufzeit der Engine in die angezeigten Seiten einzubinden. Somit wurde die 100%ige Gleichheit dieser Bereiche gewährleistet; auch, wenn abteilungsfremde Mitarbeiter u. U. Inhalte ändern mussten.

Die Frage der seiten-abhängigen Elemente war etwas schwieriger zu lösen. Als Datenquelle kamen drei Möglichkeiten in Frage:

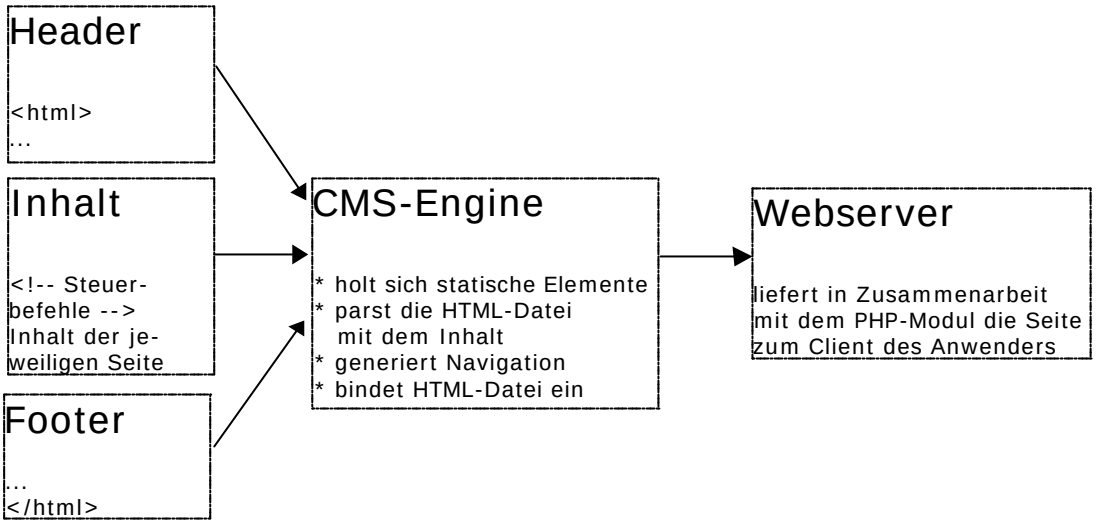
- in einer Datenbank ausgelagerte Daten, die mit einem PHP-Skript in die Seite eingebunden werden konnten,
- seperate HTML-Dateien, die nach Bedarf miteingebunden werden sollten und
- seperate XML[20]-Dateien, die genauso eingebunden werden sollten, zusätzlich jedoch noch Steuerinformationen beinhalten konnten (dazu mehr weiter unten bei den Erweiterungsmöglichkeiten).

Der Anforderungskatalog sah wieder eine hohe Performanz und Sicherheit der Lösung vor. Deshalb verzichteten wir auf eine direkte Datenbankanbindung und ließen das Skript Dateien integrieren. XML als Datenformat erschien sehr interessant, vor allem in Verbindung mit RSS[21], einer speziell für CMS entwickelter XML-Definition, mit dessen Hilfe ein späterer Umstieg auf ein professionelles CMS möglich würde. Leider erwies sich die XML-Lösung als schwer umsetzbar, da ein XML-Parser von der vorliegenden PHP-Version noch nicht unterstützt wurde. Der Aufwand einen

syntaktisch richtigen XML-Parser selbst zu entwickeln ist hoch, deshalb entschieden wir uns eine HTML-Datei dynamisch zu includen. Falls die Seite Daten beinhaltete, die sich ändern konnten (also Produktfeatures und -preise) wurden ihr PHP-Befehle hinzugefügt, die ein CSV parsten und die entsprechenden Werte ausgaben. Der Navigationskasten wurde im ersten Ansatz der Engine ebenfalls als HTML-Datei eingebunden. Aufgrund der bereits bekannten Problematik entschied ich mich ihn dynamisch zu erzeugen. Dazu wird eine zusätzliche Definitionsdatei eingebunden. In dieser ist die komplette Navigationsstruktur der Website als dreidimensionales Array vorhanden. Ein Beispiel soll dies verdeutlichen:

```
$navigation[1][0][0] = 'Kapitel 1';
$navigation[1][1][0] = 'Unterpunkt 1';
$navigation[1][1][1] = 'Seite 1';
$navigation[1][1][2] = 'Seite 2';
$navigation[1][2][0] = 'Unterpunkt 2';
...
```

Ein Kommentar in der jeweiligen Inhalts-HTML-Datei sollte den Zustand des Navigationsmenüs definieren. Die Funktion `show_navigation()` erstellt dann aus diesen Daten den Navigationskasten.



Bei der Durchführung wurden zuerst die HTML-Dateien vom sehr schlechten FrontPage-Code befreit. Sämliche Seitenaufrufe werden durch die Datei `index.php3` bearbeitet. Als erster Schritt wird der Header eingebunden. Die Engine erwartet als Argument den Namen der einzubindenden Datei. Ohne Angabe eines Parameters wird die Startseite der Präsentation geladen. Mit der `fgets()` wird die erste Zeile der Datei eingelesen. Hier befindet sich die Information, wie der Navigationskasten aufgebaut sein muss. Nach dem Rendern der Navigation wird die Inhaltsseite includet. Als Seitenabschluss wird der Footer ausgegeben.

Die bisherige Herangehensweise führte zu einem dynamischen CMS, das bei jedem Aufruf erneut die HTML-Seiten zusammenstellt. Besser wäre ein statisches CMS gewesen, dass einmal -angestossen durch den Aufruf eines speziellen Skripts- die anzuzeigenden Seiten generiert und auf dem Webserver speichert. Die Anwender greifen dann nur noch auf die erstellten Seiten zu. Nachteil dieser Lösung ist nicht absolute Akutalität und der zusätzliche Arbeitsschritt der Erzeugung der Dateien. Als gravierender Vorteil ist jedoch die geringere Prozessorlast zu sehen, auf die man vor allem auf einem Shared Hosting Account (also einem, der sich den Platz auf dem Server u. U. mit einigen tausend anderen teilt) unbedingt achten sollte.

Selbstverständlich wurden hier die Fähigkeiten eines echten CMS noch lange nicht erreicht: Noch nicht implementiert war ein Zugriffsschutz auf Schreibebene. Bisher konnte jeder, der FTP[22]-Schreibrechte besaß die Inhalte ändern. Eine Abhilfe wäre die radikale Einschränkung des FTP-

Zugriffs und ein PHP-Frontend mit Authentifizierung des Benutzers. Falls dieser das Recht hat den Inhalt zu ändern, wird ihm eine entsprechende Maske angeboten. Seine Eingaben werden in der einzubindenden Datei gespeichert und stehen dem Publikum zur Verfügung.

Interessant wäre auch die Möglichkeit Seiten vorzubereiten und ein Veröffentlichungsdatum zu bestimmen (analog zu den Kunden-werben-Kunden Seiten). Dadurch liesse sich ein "Im-Vorraus-Arbeiten" realisieren.

Ein weiterer, aus Gründen der Website-Optimierung höchst interessanter Punkt wäre die Erweiterung des Modells um ein Modul zum user-tracking mittels Sessions. Dabei kann der Weg des einzelnen Benutzers über die Site genau verfolgt werden. Durch den Einsatz von Cookies kann man herausfinden, ob der Benutzer quasi ein "Stammgast" ist und immer wieder die gleichen Seiten ansteuert oder ob er nur "Laufkundschaft" darstellt. Ebenfalls wird genaues Analysieren der Anwesenheitsdauer des Gastes möglich. Die Qualität der Daten ist beim user-tracking mit sessions erheblich genauer als beim Betrachten der sehr umfangreichen und unübersichtlichen Webserver-Logbuchdateien. Leider sind die Session-Funktionen von PHP erst ab der Version 4.0, mit den phpLib-Extensions ab PHP 3.0.12 verfügbar. Eine Nachrüstung der Tracking-Funktionalität scheint jedoch machbar.

4.Weitere Aufgaben

Neben diesen umfangreichen Projekten waren noch zahlreiche andere Aufgaben zu erledigen. Auch auf diese soll hier kurz eingegangen werden:

4.1.Auswahl und Erstellung von HTML-Templates

Verio bietet seinen Kunden die Erstellung von Internetpräsenzen an. Dabei werden vorbereitete Templates benutzt. Meine Aufgabe war es, mit Dreamweaver neue Templates zu erstellen und alte, nicht mehr zeitgemässe auszusortieren. Gelegentlich wurde ich auch mit der Aufgabe der Template-Anpassung für Kunden betraut.

4.2.Internet-Recherche über neue Produkte

Von Zeit zu Zeit wurde ich damit beauftragt, mich über neue Produkte zu informieren und damit Investitionsratschläge für die Abteilung zu geben. Meine Recherchen führten zu einigen Kaufentscheidungen. Dies waren sowohl Software wie Adobe PhotoShop 6 und UltraEdit 7.2 als auch Hardware wie z. B. der Scanner Epson Perfection 1200.

4.3.Erstellen von Grafiken

Als letzte nicht programmierbezogene Arbeit bleibt noch das Erstellen von diversen Grafiken aufzuführen. Diese wurden hauptsächlich auf der Website www.www-service.de verwendet. Auch das Gestalten von Bannern für unsere AdServer-Werbeschaltungen fiel in meinen Aufgabenbereich. Zum Gestalten der Grafiken wurden Adobe PhotoShop und Illustrator benutzt.

5.Persönliche Meinung

Abschließend bleibt noch der sehr positive Eindruck von der Arbeitsatmosphäre in einem jungen, dynamischen Unternehmen zu erwähnen. Das Klima unter den Kollegen war hervorragend. Trotzdem wurden die Projekte leider durch unzureichende Kommunikation zwischen den Abteilungen und innerhalb der Abteilung behindert. Auch exaktere Schnittstellendefinitionen hätte im Rückblick betrachtet viele Probleme von vorne herein ausgeschlossen. Wie ich in einem Gespräch mit meinem Praktikumsbetreuer nach Abschluss meines Praktikums erfahren habe hat das Unternehmen die Arbeiten zu einer Zertifizierung nach DIN EN ISO 9001. Diese Norm garantiert Qualitätssicherung und wird Arbeitsabläufe standardisieren.

Als sehr wichtig und interessant empfand ich die Möglichkeit an firmeninternen Schulungen (z. B. zum hausintern entwickelten Fakturierungsprogramm oder Outlook) teilzunehmen.

Bemerkenswert waren die Einblicke in die serverbasierte Programmierung und deren Möglichkeiten. Diese Form der Web-Applikationen nimmt bereits einen grossen Raum in der IT[23]-Wirtschaft ein und ist zuküftig sicher nicht

mehr wegzudenken.

Zu guter Letzt möchte ich meinem Praktikumsbetreuer Hannes Klessinger für die stete Unterstützung herzlich danken.

6.Abkürzungsverzeichnis

Im Bericht wurden folgende Abkürzungen verwendet. Die Zahlen in Klammern geben die hochgestellte Zahl im Text an.

- Blob Binary Large Object: Datenformat innerhalb mySQL (11)
- CMS Content Management System: Verwaltungsprogramm für Webseiten (18)
- CSS Cascading Style Sheet: Formatierungsanweisungen für HTML-Seiten (14)
- CSV Comma Seperated Value: Datenformat, dass die Werte mit Kommas voneinander trennt. (17)
- EDV Elektronische Datenverarbeitung: Verwaltung von Informationen durch Computertechnik (8)
- FTP File Transfer Protocol: Protokoll, dass den Transfer von Dateien im Internet regelt. (22)
- GIF Graphics Interchange Format: verlustfreies Komprimierungsverfahren für Bilddaten (9)
- GPL GNU Public License: freies Lizenzmodel (19)
- HTML Hypertext Markup Language: Spezielle Auszeichnungssprache für grafische Internetseiten (15)
- HTTP Hypertext Transfer Protocol: Protokoll, dass der Übertragung von z. B. HTML-Seiten zugrundeliegt. (3)
- HTTPS Hypertext Transfer Protocol Security: Protokoll, dass zusätzlich Verschlüsslungs- und Authentifizierungsmöglichkeiten bietet. (4)
- IT Information Technology: Bezeichnung für den informationsverarbeitenden Wirtschaftssektor. (23)
- JPEG Joint Photographic Experts Group: verlustbehaftetes Komprimierungsverfahren für Bilddaten (10)
- NTT Nippon Telephone & Telegraph Public Corporation:

japanischer Telekommunikationskonzern (1)

- PGP Pretty Good Privacy: sehr sicheres Verschlüsslungs-programm, das mit einem asymetrischen Verfahren arbeitet. (16)
- PHP PHP Hypertext Processor: serverseitige Skriptsprache (2)
- RDBMS Relational Data Base Management System: Organisationsmodell für Datenbanken (5)

RSS Rich Site Summary: Standard für Content Management per XML. Spezifikation siehe: http://backend.userland.com/rss091/ (21)

- SQL Structured Query Language: Abfragesprache für Datenbanken (6)
- URL Uniform Resource Locator: eindeutige Adresse für Daten im Internet (13)
- XML Extended Markup Language: universelle Auszeichnungssprache. Spezifikation siehe: http://www.w3.org/XML/ (20)

7.Literaturverzeichnis

Bei der Erstellung dieses Berichts und im Laufe des Praktikums wurden folgende Informationsquellen benutzt. Die Zahlen in Klammern korrespondieren wieder mit den hochgestellten Zahlen im Bericht.

7.1.Print-Medien

- Autor unbekannt,
  Skript zur praktikumsbegleitenden Lehrveranstaltung „kom-merzielle Programmierung“,
  ohne Ort, ohne Erscheinungsjahr,
  Seite L1-1
  (7)

- Christian Wenz,
  „JavaScript - Browserübergreifende Lösungen“,
  ISBN: 393435894,
  Gallileo Press,

Bonn, 2000

- Jörg Krause,
  „PHP 4. Grundlagen und Profiwissen“,
  ISBN: 3446215468,
  Carl Hanser Verlag,
  München, 2000
  (2a)

7.2.Online-Medien

- Autor unbekannt, offizielle PHP-Dokumentation,
  www.php.net
- Autor unbekannt, offizielle mySQL-Dokumentation,
  www.mysql.com

- Stefan Münz, SelfHTML,
  Version 7.0 vom 27. April 1998,
  www.teamone.de/selfhtml
  (14a)
- zahlreiche Autoren, PHP Mailingliste,
  zu abonieren über infosoc.uni-koeln.de/mailman/listinfo/php
  (12)