jenigen, die bislang noch nicht mit Linux arbeiten, von LAMP überzeugt werden und so den Weg zu Linux finden.

1.4 Beispiel für den fortgeschrittenen Benutzer

Im Folgenden soll ein Beispiel erläutert werden, das die wichtigsten Funktionen und Bestandteile eines LAMP-Systems verwendet. Dieses Beispiel ist für den bereits erfahreneren Benutzer gedacht, der schnell Ergebnisse sehen möchte, ohne sich zuvor das Hintergrundwissen anzueignen, das in den nachfolgenden Kapiteln vermittelt wird.

Falls Sie neu in die Materie LAMP einsteigen, sollten Sie den Rest dieses Kapitels überspringen und direkt mit Kapitel 2 beginnen. Nachdem Sie die anderen Kapitel über Linux, Apache, MySQL und PHP gelesen haben, können Sie sich nochmals diesem Beispiel als Fallstudie widmen.

In den weiteren Kapiteln werden andere Beispiele verwendet, um unterschiedliche Fälle aus der Praxis und Herangehensweisen an diese Thematik kennen zu lernen.

Das erste Beispiel ist eine kleine Anwendung für einen Pizzabäcker. Hier wird dem Kunden eine Webseite präsentiert, die es ihm ermöglichen soll, eine Pizza zu bestellen. Das allein ist natürlich relativ trivial, daher soll im Verlauf dieses Beispiels diese Bestellung immer weiter verfeinert werden, um dann beispielsweise die Pizza komplett belegen zu können. Auf der anderen Seite muss unser Pizzabäcker auch die Daten der Pizzen bzw. die Zutaten verändern können. Dafür soll ebenfalls ein eigener Bereich geschaffen werden, der nur für den Pizzabäcker erreichbar ist.

1.5 Das Beispiel „Pizzaverkauf“

1.5.1 Vorbereitung des Systems

Als Vorbereitungen für die Beispiele benötigen Sie natürlich ein lauffähiges Linux-System. Im weiteren Verlauf wird davon ausgegangen, dass dies ein SuSE Linux ist und Sie die Grundinstallation bereits durchgeführt haben.

Darüber hinaus sollten noch die verschiedenen Komponenten, die für ein LAMP-System benötigt werden, installiert werden. Diese Komponenten (Pakete) heißen bei SuSE Linux ab Version 7.1 (die Pakete sind zumeist auch auf älteren Versionen enthalten, aber möglicherweise in anderen Serien oder unter anderem Namen abgelegt) wie folgt:

Apache – Paketname: `apache`
Serie: `n`
Gruppe (SuSE Linux $<$ 8): `Netzwerk/Daemonen`
Gruppe (SuSE Linux $\geq$ 8): `Produktivität/Netzwerk/Web/Server`

MySQL – Paketname: `mysql`
Serie: `ap`
Gruppe (SuSE Linux $<$ 8): `Anwendungen/Datenbanken`
Gruppe (SuSE Linux $\geq$ 8): `Produktivität/Datenbanken/Server`

PHP4 – Paketname: `mod_php4`, `mod_php4_core`
Serie: `n`
Gruppe (SuSE Linux $<$ 8): `Netzwerk/Daemonen`
Gruppe (SuSE Linux $\geq$ 8): `Produktivität/Netzwerk/Web/Server`

Zusätzlich zu den oben genannten Paketen können Sie die Funktionalität Ihres LAMP-Systems noch mit den unten aufgeführten Komponenten erweitern. Besonders wichtig (aber eben nicht unbedingt erforderlich) ist dabei `phpMyAdmin` (dieses Paket ist seit Version 7.3 enthalten). Mit diesem Tool können Sie sehr einfach administrative Arbeiten an Ihrer Datenbank durchführen (siehe Kapitel 6).

In der Apache-Dokumentation sind die Konfigurationsdirektiven für Apache als HTML-Seiten nachzulesen (siehe B.1.1).

Auch die PHP-Dokumentation finden Sie als HTML-Seiten unter den optionalen Komponenten.

Um in Ihrer Datenbank einige der neuen Funktionen von MySQL verwenden zu können, installieren Sie das Paket `mysql-max` (dies ist als Paket in SuSE Linux seit Version 7.3 enthalten). Diese neuen Funktionen umfassen unter anderem Transaktionen mit dem Tabellentyp Berkley DB (BDB) (siehe 4.10.5) und auch Replikationen mit anderen Servern. Sobald Sie dieses Paket installiert haben, wird der Standard-MySQL-Dämon automatisch durch den Dämon ersetzt, der diese Funktionen zur Verfügung stellt.

Durch den MySQL-Client sind Sie in der Lage, mit einer textbasierenden Konsole SQL-Kommandos an Ihren MySQL-Server zu senden. Die Funktionsweise des MySQL-Clients wird im Abschnitt 4.7 genauer beschrieben.

Apache Dokumentation – Paketname: `apache-doc`
Serie: `n`
Gruppe (SuSE Linux $<$ 8): `Netzwerk/Daemonen`
Gruppe (SuSE Linux $\geq$ 8): `Produktivität/Netzwerk/Web/Server`

MySQL Client – Paketname: `mysql`
Serie: `ap`
Gruppe (SuSE Linux < 8): `Anwendungen/Datenbanken`
Gruppe (SuSE Linux ≥ 8): `Produktivität/Datenbanken/Clients`

MySQL Server mit Transaktionen – Paketname: `mysql-max`
Serie: `ap`
Gruppe (SuSE Linux < 8): `Anwendungen/Datenbanken`
Gruppe (SuSE Linux ≥ 8): `Produktivität/Datenbanken/Server`

phpMyAdmin – Paketname: `phpMyAdmin -` Serie: `n`
Gruppe (SuSE Linux < 8): `Anwendungen/Netzwerk`
Gruppe (SuSE Linux ≥ 8): `Produktivität/Netzwerk/Web/Frontends`

PHP-Dokumentation – Paketname: `phpdoc`
Serie: `doc`
Gruppe (SuSE Linux < 8): `Dokumentation`
Gruppe (SuSE Linux ≥ 8): `Produktivität/Netzwerk/Web/Server`

Nachdem Sie die Pflichtpakete und idealerweise auch gleich die optionalen Pakete installiert haben, müssen Sie Linux nun noch mitteilen, dass Apache und MySQL immer gestartet werden.

Falls Sie SuSE Linux bis Version 7.3 verwenden, erledigen Sie dies mit der SuSE-eigenen Konfigurationsdatei `rc.config` wie nachfolgend beschrieben:

Editieren Sie hierzu als `root` die Datei `rc.config` (entweder mit einem Texteditor wie `vi` oder mit dem in YaST 2 integrierten `RC.Config-Editor`). Setzen Sie die benötigten Variablen wie folgt:

```
START_HTTPD=yes
START_MYSQL=yes
```

Mit Version 8.0 hat sich der Start von Serverdiensten grundlegend geändert, denn nun wird die komplette Konfiguration im Verzeichnis `/etc/sysconfig` abgelegt. Um hier den Webserver und die Datenbank ab dem nächsten Systemstart automatisch starten zu lassen, geben Sie in einer Konsole als `root` folgende Befehle ein:

```
linux: # insserv apache
linux: # insserv mysql
```

Alternativ dazu können Sie auch mit YaST 2 unter *System* den *Runlevel-Editor* verwenden. Eine Beschreibung dazu finden Sie in den Kapiteln 3 und 4.

Um sofort mit den beiden Dämonen arbeiten zu können, haben Sie unter SuSE Linux die Möglichkeit, diese über die so genannten rc-Skripte zu starten. Apache wird über folgenden Aufruf gestartet:

```
linux: # rcapache start
Starting httpd [ PHP4 ]                                          done
```

Beim Starten von Apache sehen Sie auch die geladenen Module zu Apache in den eckigen Klammern. Im obigen Fall wird das PHP 4-Modul zu Apache bereits hinzugeladen. Für Sie heißt das, dass sowohl die Installation von Apache als auch die Installation von PHP4 erfolgreich war. Genauere Informationen zur Aktivierung und zum Start von Apache können Sie im Abschnitt 3.3.4 nachlesen.

Der Start von MySQL läuft prinzipiell nach dem gleichen Muster. Das heisst, dass Sie auch hier mit dem rc-Skript den eigentlichen Dämon starten. Allerdings sollte nach dem Start von MySQL noch ein Passwort für den Datenbankadministrator festgesetzt werden (dies erscheint auch als Textinformation beim ersten Start von MySQL). Der Ablauf des Startens von MySQL und der Passwortänderung wird im Abschnitt 4.6.2 noch genauer erläutert.

```
linux: # rcmysql start
Creating MySQL privilege database and starting MySQL
...
PLEASE REMEMBER TO SET A PASSWORD FOR THE MySQL root USER !
This is done with:
/usr/bin/mysqladmin -u root -p password 'new-password'
/usr/bin/mysqladmin -u root -h linux -p password 'new-password'
See the manual for more instructions.
...
```

Für den Moment genügt es, wenn Sie den Anweisungen der Textinformationen folgen und hier folgende zwei Kommandos als `root` eingeben:

```
linux: # mysqladmin -u root -p password 'neuespasswort'
linux: # mysqladmin -u root -h linux -p password 'neuespasswort'
```

Ersetzen Sie bei den beiden Kommandos oben den Text `neuespasswort` mit Ihrem gewünschten Passwort für den Datenbankadministrator. Sie müssen zusätzlich noch den Rechnernamen Ihres Linux-Rechners in der zweiten Zeile statt des Textes `linux` eintragen. Achtung: Mit einigen SuSE-Linux-Versionen (unter anderem auch 8.0) wird diese Änderung nicht auf Anhieb funktionieren. In der Datenbank steht als Host eingetragen `linux`, während mit dem Kommando `mysqladmin` als Host `linux.local` mitgeliefert wird. Da es allerdings völlig ausreicht, von `localhost` einen Zugriff zu haben, müssen Sie sich momentan keine weiteren Gedanken machen. Nur im produktiven Einsatz sollte dieser Eintrag nicht ohne Passwort stehen bleiben oder am besten komplett entfernt werden. Lesen Sie dazu den Abschnitt 4.12.1.

Damit sind die ersten Hürden zum LAMP-System bereits genommen. Alle wichtigen Bestandteile sind lauffähig. Nun müssen Sie die Veröffentlichung Ihrer dynamischen Webseiten vorbereiten.

1.5.2 Vorbereitungen unter Linux

Im nächsten Teil der Vorbereitungen geht es nun um konkrete Benutzer-Einstellungen unter Linux. Damit das Pizzaprojekt ungestört von sonstigen Aktivitäten auf Ihrem Rechner laufen kann, sollten Sie zunächst einen eigenen Benutzer für dieses Projekt anlegen. Dies kann am einfachsten mit YaST 2 geschehen (siehe 2.4.2) oder aber auch mit dem Kommandozeilenprogramm `useradd`. Legen Sie mit einer der beiden Methoden einen Benutzer `pizza` an.

Im folgenden Beispiel für die Verwendung von `useradd` werden die Parameter `-m` und `-c` benutzt. Der Parameter `-m` bewirkt, dass das Home-Verzeichnis des Pizzabäckers mit angelegt wird und einige wichtige Grunddateien in dieses Verzeichnis bereits kopiert werden. Die Angabe `-c` kann verwendet werden, um den vollständigen Namen, Büronummer etc. anzugeben. Hier wird als Kommentar der Name `Pizzabäcker` eingetragen.

```
linux: # useradd -m -c "Pizzabäcker" pizza
linux: # passwd pizza
New Password:
Re-enter new password:
Password changed
```

Sie könnten bei `useradd` auch gleich ein Passwort mit dem Parameter `-p` übermitteln. Dieses Passwort müsste allerdings bereits in verschlüsselter Form angegeben werden. Um dies zu vermeiden, wird an der nächsten Kommandozeile mit `passwd pizza` das Passwort für den User `pizza` eingestellt. Weisen Sie dem Benuzter `pizza` ein Passwort zu (verwenden Sie kein leeres Passwort, da einige Systemdienste nur mit einem eingetragenen Passwort funktionieren).

Die Grundeinstellungen für die Berechtigungen des Home-Verzeichnisses dieses Benutzers hängen allerdings davon ab, wie Sie den Benutzer angelegt haben. Falls Sie YaST 2 verwendet haben, so ist das Home-Verzeichnis standardmäßig nur für den Benutzer selbst freigegeben. Da Sie aber Seiten ins World Wide Web stellen möchten, muss auch die Gruppe `Sonstige` auf dieses Verzeichnis Zugriff haben. Dies kann wieder auf verschiedene Arten erledigt werden: Sie können entweder mit dem Kommando `chmod` die Berechtigungen setzen. Falls Sie sich als Benutzer `pizza` bereits unter KDE angemeldet haben, können Sie alternativ dazu auch mit dem *Konqueror* die Berechtigungen verändern. Setzen Sie als Mindesteinstellung die Berechtigung `x` (heißt im *Konqueror „Öffnen“*) für die Gruppe `Sonstige`.

Hier soll als Beispiel `root` das Kommando `chmod` verwenden, um das Home-Verzeichnis des Benutzers `pizza` anzupassen:

```
linux: # chmod o+x ~pizza
```

Der Parameter `o+x` bewirkt, dass für `Sonstige` die Berechtigung `x` hinzugefügt wird und alle anderen bereits bestehenden Berechtigungen unverändert bleiben. Eine genauere Beschreibung zu `chmod` finden Sie im Abschnitt 2.5.1. Die Angabe `~pizza` kann verwendet werden, um das Home-Verzeichnis eines bestimmten Benutzers zu kennzeichnen. Direkt im Anschluss an die Tilde wird der Benutzername notiert. Dadurch können Sie von jeder beliebigen Stelle im Dateisystem auf die Home-Verzeichnisse der User zugreifen. Alternativ dazu hätten Sie natürlich auch den kompletten Pfad `/home/pizza` schreiben können.

1.5.3 Die ersten Webseiten

Nun sollten Sie sich als Benutzer `pizza` am System anmelden, um hier lokale Arbeiten durchführen zu können (administrative Aufgaben können ja über eine `root`-Konsole weiterhin erledigt werden).

Nach der Anmeldung testen Sie zunächst einmal, ob Apache funktioniert und auch für den Benutzer `pizza` ein eigenes Verzeichnis zur Verfügung steht. Dies wird am besten mit dem *Konqueror* als Web-Browser (oder auch mit jedem anderen Browser) durchgeführt. Starten Sie den Web-Browser und geben Sie folgenden URL ein:

```
http://localhost/~pizza/
```

Durch diesen Aufruf sollte ein leeres Verzeichnis im Browser angezeigt werden (siehe Abbildung 1.1).

Abbildung 1.1: *Konqueror* mit leerem Verzeichnis

Dieses Verzeichniss soll nun mit einer ersten HTML-Seite gefüllt werden. Dazu müssen Sie zuerst einmal wissen, an welcher Stelle diese Webseite im lokalen Dateisystem abzulegen ist. URLs, bei denen nach dem Rechnernamen (hier `localhost`) ein Benutzername mit vorangestellter Tilde (`~pizza`) angegeben ist, suchen im Home-Verzeichnis des angegebenen Benutzers ein Unterverzeichnis `public_html` und zeigen die darin befindlichen Dateien an (siehe Abschnitte 3.3.3 und 3.3.8). Im aktuellen Fall hat der User `pizza` in seinem Home-Verzeichnis auch tatsächlich ein Unterverzeichnis `public_html`. Dieses Verzeichnis wurde beim Anlegen des Benutzers mit YaST 2 oder mit `useradd` automatisch angelegt.

Wechseln Sie nun (als Benutzer `pizza`) in dieses Verzeichnis `public_html` und erzeugen Sie darin eine neue Datei `willkommen.html` mit einem beliebigen Texteditor. Der Inhalt dieser Datei sollte in etwa wie folgt aussehen:

```
<html>
<head><title>Pizzabäcker Bestellservice</title></head>
<body>
<h1 align="center">Herzlich Willkommen</h1>
<h2 align="center">zum</h2>
<h1 align="center">Pizzabäcker Bestellservice</h1>
<h3 align="center">
    <a href="bestellung.php">Pizzabestellung</a>
</h3>
</body>
</html>
```

Nach dem Speichern dieser Seite aktualisieren Sie im (hoffentlich noch geöffneten) Web-Browser die bisher leere Seite des Benutzers `pizza`. Es ist nun ein Link auf die Seite `willkommen.html` vorhanden. Zum Testen dieser Seite klicken Sie diesen Link an. Damit sollten Sie Ihre erste Seite für den Pizzabäcker bereits festgelegt haben. Falls Sie nun noch erreichen möchten, dass diese Seite als Standardseite beim Zugriff auf den URL `http://localhost/~pizza/` verwendet wird, so sollten Sie diese Seite in `index.html` umbenennen. Dieser Dateiname ist in der Apache-Direktive `DirectoryIndex` als Standard für einen direkten Verzeichnisaufruf definiert (siehe Abschnitt B.1.1). Benennen Sie die Seite entweder mit dem *Konqueror* oder mit folgendem Kommando in der Konsole um:

```
pizza@linux:~/public_html > mv willkommen.html index.html
```

Der vorhande Link auf die Seite `bestellung.php` kann natürlich noch nicht funktionieren, da Sie diese Seite erst einmal anlegen müssen. Die wichtigste Frage für die Bestellseite sollte dabei sein: Was kann bzw. muss der Kunde alles eingeben?

Folgende Informationen sollen zuerst einmal für das Beispiel eingegeben werden:

- Anrede
- Vorname
- Name
- Straße
- PLZ
- Ort
- E-Mail
- Gewünschte Zutaten für die Pizza

Diese Bestellseite wird also ein Formular, das z. B. folgendermaßen aussehen könnte (speichern Sie dieses Formular unter dem Namen `bestellung.php`).

```
<html>
<head><title>Pizzabäcker Bestellseite</title></head>
<body>
<h3>Pizzabestellung</h3>
<form action="bestellung-2.php">
<table>
<tr>
<td>Anrede</td>
<td><select name="anrede"><option>Herr<option>Frau</td>
</tr>
<tr>
<td>Vorname</td>
<td><input type="text" name="vorname"></td>
</tr>
<tr>
<td>Name</td>
<td><input type="text" name="name"></td>
</tr>
<tr>
<td>Straße</td>
<td><input type="text" name="strasse"></td>
</tr>
<tr>
<td>PLZ Ort</td>
<td><input type="text" name="plz" size="9">
    <input type="text" name="ort"></td>
</tr>
<tr>
<td>E-Mail</td>
<td><input type="text" name="email"></td>
</tr>
```

```
<tr>

<td>Zutaten</td>
<td><input type="checkbox" name="zutaten[]" value="Salami">
    Salami<br>
    <input type="checkbox" name="zutaten[]" value="Schinken">
    Schinken<br>
    <input type="checkbox" name="zutaten[]" value="Champignons">
    Champignons<br>
    <input type="checkbox" name="zutaten[]" value="Peperoni">
    Peperoni<br>
    <input type="checkbox" name="zutaten[]" value="Paprika">
    Paprika<br>
</td>
</tr>
</table>
<input type="submit" value="bestellen">
</form>
</body>
</html>
```

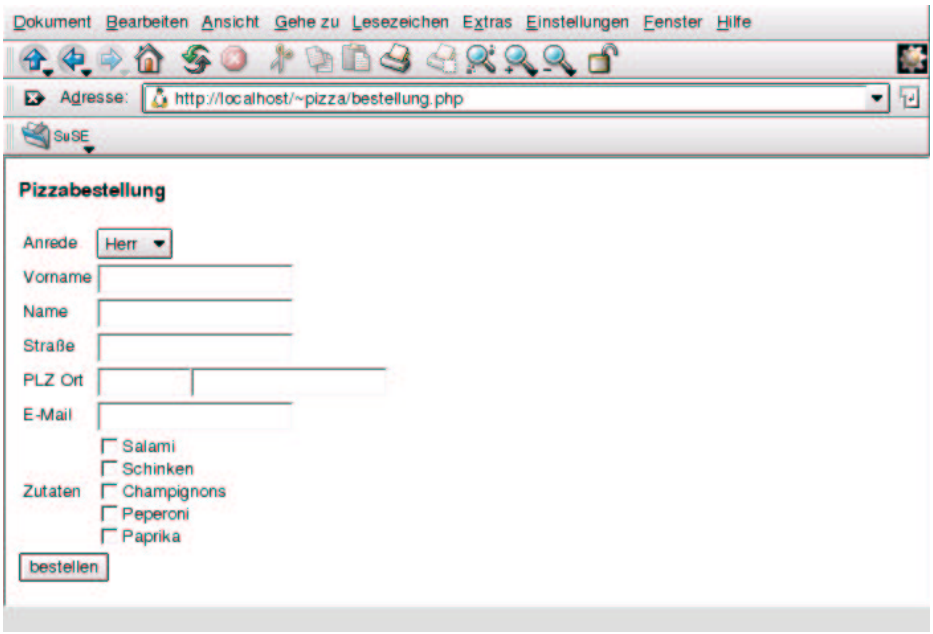

Abbildung 1.2: *Konqueror* mit der Bestellseite

Auf der Bestellseite, die Sie in Abbildung 1.2 sehen, kann nun der Benutzer seine persönlichen Daten angeben und im unteren Bereich die Zutaten bestimmen, mit denen die Pizza belegt werden soll. Beim Formular ist auf Folgendes zu achten:

- Beim Beginn des Formulars (`<form>`-Tag) muss angegeben werden, an welches Skript die eingegebenen Daten gesendet werden sollen (der Name hinter `action`).
- Zur übersichtlicheren Darstellung wird das Formular in eine Tabelle eingepackt.
- Jedes Eingabefeld muss einen eindeutigen `name` bekommen. Dieser Feldname im Formular wird in PHP anschließend zum Variablennamen (siehe Abschnitt 5.6.5.4).
- Durch Anhängen der eckigen Klammern (`[]`) an den Feldnamen können Sie auch mehreren Feldern (oben sind das die Zutaten) den gleichen Namen geben. Die angewählten Informationen werden in der Auswertung in ein Array gespeichert (siehe Abschnitt 5.6.4).

Eventuell haben Sie sich bereits gefragt, warum diese Seite überhaupt mit der Endung `.php` gespeichert wurde und nicht mit `.html`, obwohl keinerlei PHP-Code auf der Seite zu erkennen ist. Die Endung soll hier nur bereits als Vorbereitung für einen später noch zu ergänzenden PHP-Teil in dieser Seite dienen, um nicht alle Links auf diese Seite später wieder ändern zu müssen. Da PHP-Seiten vom Grundgerüst her wie normale HTML-Seiten betrachtet werden und nur an speziell gekennzeichneten Stellen die PHP-Funktion wirksam wird, wird im aktuellen Fall diese Seite auch als normale HTML-Seite an den Browser geschickt.

Mit dieser Ergänzung ist nun bereits die erste Webseite eines einfachen Bestellsystems für den Pizzabäcker eingerichtet. Weitere Ergänzungen betreffen nun den Komfort bei der Verwaltung der Pizzazutaten und der eingegangenen Bestellungen. Zudem sollen noch Sicherheitsfunktionen für die administrativen Seiten und ein eigener virtueller Host für die Pizzabestellungen eingerichtet werden.

1.5.4 Ein virtueller Host für den Pizzabäcker

Im Moment ist die Seite des Benutzers `pizza` nur über den etwas unschönen URL `http://localhost/~pizza/` (oder natürlich auch über die IP-Adresse Ihres Rechners) zu erreichen. Der Webserver Apache unterstützt die Möglichkeit, sog. *virtuelle Hosts* auf einem Server zur Verfügung zu stellen. Im Folgenden soll nun ein solcher virtueller Host unter dem Namen `web-pizza.local` eingerichtet werden.

Dies erfordert einige Schritte, die als `root` durchgeführt werden müssen. Zuerst bedienen Sie sich eines kleinen Tricks, um den neuen Namen (ohne DNS zu verwenden) `web-pizza.local` auf Ihrem Linux-Rechner bekannt zu machen. Edi-

tieren Sie die Datei /etc/hosts und tragen Sie folgende Zeile am Ende dieser Datei ein:

```
192.168.1.201   web-pizza.local         web-pizza
```

Ersetzen Sie die IP-Adresse 192.168.1.201 durch die IP-Adresse Ihres Linux-Rechners. Dadurch können Sie lokal von Ihrem Rechner den „neuen Rechner“ web-pizza.local erreichen.

Im zweiten Schritt müssen Sie Apache so konfigurieren, dass ein Zugriff auf die Domain web-pizza.local auf die Startseite des Pizzabäckers umgeleitet wird.

Hierfür gibt es zwei Möglichkeiten, um diesen virtuellen Host in Apache zu integrieren: Sie können eine mit SuSEconfig verträgliche Variante wählen; auf diese Weise können Sie nach wie vor einige Einstellungen mit Variablen in Konfigurationsdateien vornehmen und damit die Apache-Konfiguration festlegen. Alternativ dazu ist es natürlich möglich, diese Einstellungen direkt in der Apache-Konfigurationsdatei httpd.conf vorzunehmen.

Falls Sie nicht mit SuSEconfig arbeiten möchten bzw. kein SuSE Linux verwenden, tragen Sie direkt in der Datei /etc/httpd/httpd.conf (oder wo auch immer Ihre httpd.conf abgelegt ist) die Direktiven ein, die unten als eigenständige Datei vhosts.conf beschrieben sind.

Für einen SuSEconfig-kompatiblen Eintrag gehen Sie wie folgt vor:

Bearbeiten Sie zuerst die Datei /etc/rc.config.d/apache.rc.config bzw. ab SuSE Linux 8.0 die Datei /etc/sysconfig/apache. Diese Datei enthält einige Konfigurationsvariablen, die SuSEconfig verwendet, um die eigentliche Apache-Konfigurationsdatei /etc/httpd/httpd.conf zu erzeugen. Eine nähere Erläuterung der einzelnen Variablen, die Sie in der SuSE-eigenen Datei apache.rc.config bzw. /etc/sysconfig/apache setzen können, finden Sie im Abschnitt 3.3.3.

Für den Augenblick ergänzen Sie die Variable HTTPD_CONF_INCLUDE_FILES um den Eintrag /etc/httpd/vhosts.conf. Falls Sie bereits einen Eintrag (oder auch mehrere) in dieser Variablen haben, geben Sie nach dem letzten Eintrag noch ein Leerzeichen ein und hängen dann den obigen Wert an. Der Eintrag in der Datei apache.rc.config bzw. /etc/sysconfig/apache sieht im einfachsten Fall nun wie folgt aus:

```
HTTPD_CONF_INCLUDE_FILES="/etc/httpd/vhosts.conf"
```

Durch die Angabe von /etc/httpd/vhosts.conf wie oben beschrieben veranlassen Sie SuSEconfig dazu, dass der Apache-Konfigurationsdatei ein Eintrag hinzugefügt wird, der die genannte Datei hinzulädt. In vhosts.conf bestimmen Sie die eigentlichen Einstellungen für den bzw. später auch für mehrere virtuelle(n) Host(s).

Legen Sie nun noch die Datei `/etc/httpd/vhosts.conf` als `root` mit folgendem Inhalt an:

```
NameVirtualHost 192.168.1.201

<VirtualHost 192.168.1.201>
        ServerName web-pizza.local
        ServerAdmin web.pizza@localhost
        DocumentRoot /home/pizza/public_html
        ErrorLog /home/pizza/vhost_error.log
        CustomLog /home/pizza/vhost_access.log combined
</VirtualHost>
```

Ersetzen Sie in der obigen Datei die IP-Adresse `192.168.1.201` wieder durch die IP-Adresse Ihres Rechners. Die entscheidende Zeile ist die Zeile `ServerName`: Hier geben Sie den Namen an, der auf ein anderes Startverzeichnis gelegt werden soll. Dieses Startverzeichnis wird durch die Angabe `DocumentRoot` festgelegt. Hier steht das Verzeichnis des Pizzabäckers, in dem die Startseite und die Bestellseite abgelegt sind. Die Angabe `ErrorLog` bestimmt die Datei, in die Fehlermeldungen von Apache für diesen virtuellen Host mitprotokolliert werden sollen. Mit der `CustomLog`-Angabe wird noch ein Logfile angelegt, in das die gesamten Zugriffe auf den virtuellen Host geschrieben werden (siehe Abschnitt 3.4). Eine ausführlichere Beschreibung zu virtuellen Hosts finden Sie im Abschnitt 3.5.

Diese Einstellungen müssen zum einen jetzt noch mit Hilfe von `SuSEconfig` in die allgemeine Konfigurationsdatei von Apache geschrieben werden und dann zum anderen Apache auch als aktuelle Konfiguration zugewiesen werden. Geben Sie als `root` die folgenden Kommandos ein:

```
linux: # SuSEconfig -module apache
Starting SuSEconfig, the SuSE Configuration Tool...
Running module apache only
Reading /etc/rc.config and updating the system...
Executing /sbin/conf.d/SuSEconfig.apache...
Installing new /etc/httpd/suse_include.conf
Finished.
linux: # rcapache restart
Shutting down httpd                                          done
Starting httpd [ PHP4 ]                                      done
```

Falls Sie kein `SuSEconfig` verwenden, müssen Sie nur Apache neu starten, um die Konfiguration erneut einzulesen.

Durch diese Angaben ist nun der virtuelle Host `http://web-pizza.local` von Ihrem Rechner aus erreichbar. Testen Sie diesen virtuellen Host in Ihrem Web-Browser.

1.5.5 Die ersten PHP-Seiten

Jetzt sollten Sie sich wieder der eigentlichen Web-Programmierung widmen und damit auch in PHP einsteigen. Dazu muss zunächst einmal geklärt werden, wie Sie PHP in HTML-Seiten integrieren können. Im einfachsten Fall schreiben Sie an der Stelle im HTML-Quelltext, an der PHP beginnen soll, die Zeichenkette `<?php`, und am Ende des PHP-Teils wiederum die Zeichenkette `?>`. Der darin befindliche Teil wird vom PHP-Interpreter ausgewertet und das Ergebnis dieser Auswertung an den Browser gesendet. Eine nährere Erläuterung zur Integration von PHP in HTML finden Sie im Abschnitt 5.3.

Nun sollen auf der Auswertungsseite zunächst einmal die Informationen angezeigt werden, die Sie im Formular eingetragen haben. Das folgende Skript ist nur als ein erster Schritt zur eigentlichen Bestellung zu sehen. Speichern Sie diese Seite als `bestellung-2.php` ab.

```
<html>
<head><title>Pizzabäcker Bestellservice</title></head>
<body>
<h1 align="center">Vielen Dank</h1>
Folgenden Daten wurden vom Formular übermittelt:<br>
<?= $anrede; ?><br>
<?php
echo "$vorname $name<br>\n";
echo "$strasse<br>\n";
echo "$plz $ort<br>\n";
echo "$email<br>\n";
?>
Sie möchten eine Pizza mit folgenden Zutaten:
<?php
# $zutaten ist ein Array also muss eine Schleife
# die Zutaten ausgeben:
foreach($zutaten as $z){
        echo "$z ";
}
?>
</body>
</html>
```

Im obigen Beispiel wurde bereits mehrfach zwischen dem HTML-Teil und dem PHP-Teil hin und her gewechselt. In PHP ist dies ohne weiteres beliebig oft möglich. Dies erleichtert Ihnen die Erstellung von festen HTML-Bestandteilen. Sie können diese Teile mit einem HTML-Editor erzeugen und dann nur Ihre PHP-Programmierung in der HTML-Seite ergänzen.

Der erste PHP-Teil (`<?= $anrede; ?>`) zeigt bereits, wie Sie auf sehr einfache Weise den Inhalt einer PHP-Variable in den HTML-Code integrieren können. Durch das Gleichheitszeichen (=) direkt nach dem Beginn des PHP-Teils weisen Sie PHP an, das nachfolgende Element sofort auszugeben. Direkt danach steht hier die Variable `$anrede`. Der Wert dieser Variablen wurde durch die Auswahl des Benutzers im Formular gesetzt (Sie haben im Formular bei `<select>` den Parameter `name="anrede"` festgelegt). Jede Eingabe bzw. Auswahl im Formular wird in der PHP-Seite automatisch in eine Variable geschrieben. Allerdings geschieht dies nur, solange der Parameter `register_globals` in der PHP-Konfiguration den Wert `on` hat. Sollte das nicht der Fall sein, so müssen Sie über andere Variablen den Wert auslesen. Lesen Sie dazu Abschnitt 5.16.1.

Im zweiten PHP-Teil wird deutlich mehr an den Browser ausgegeben, daher werden hier mehrere `echo` Kommandos für die Ausgabe verwendet (Sie hätten auch alles in einer Zeile schreiben können, aber dadurch verliert der Code an Übersichtlichkeit).

Der dritte PHP-Teil beginnt mit zwei Kommentarzeilen in „Shell/Perl-Schreibweise“ (das Doppelkreuz # am Zeilenanfang). Im Anschluss daran wird mit einer Kontrollstruktur (die `foreach`-Schleife wird im Abschnitt 5.7.6 erläutert) der Inhalt des Arrays `$zutaten` durchlaufen, und jede einzelne Zutat wird über die Variable `$z` an den Browser ausgegeben.

Jetzt wird es wieder Zeit, diese Seiten einmal auszuprobieren. Starten Sie vom URL `http://localhost/~pizza/` (Sie müssten jetzt bereits die „Willkommen“-Seite sehen, da diese als `index.html` abgespeichert ist). Von da aus gelangen Sie über den Link `Bestellung` zum Bestellformular. Füllen Sie das Formular aus und senden Sie Ihre Bestellung ab. Nun wird Ihre Bestellung von `bestellung-2.php` ausgewertet und Sie bekommen die einzelnen Eingaben aus dem Formular wieder angezeigt.

1.5.6 Ein dynamisches Formular

Das Bestellformular wäre für den Pizzabäcker noch etwas komfortabler, wenn für die Zutaten nicht jede einzeln als `checkbox` angelegt werden müsste, sondern wenn dies automatisch erledigt werden könnte.

Dazu muss die Seite `bestellung.php` jetzt ebenfalls mit dynamischem Inhalt erzeugt werden. Ändern Sie die Seite `bestellung.php` wie folgt ab:

Löschen Sie in der Tabelle den Inhalt der Zutatenzelle (die HTML-Tags von

```
<input type="chechbox" name="zutaten[]" value="salami">
```

bis zu

```
Paprika<br>
```

nach der letzten checkbox). In der nun leeren Zelle (<td> </td>) tragen Sie folgenden PHP-Code ein:

```
<?php
$zutaten=array( "Salami",
                "Schinken",
                "Champignons",
                "Peperoni",
                "Paprika");
foreach($zutaten as $z){
   echo "<input type=\"checkbox\" name=\"zutaten[]\"
          value=\"$z\">$z<br>\n";
}
?>
```

Mit dieser Ergänzung wird das Bestellformular dynamisch erzeugt. Die einzelnen checkbox-Einträge werden im Array $zutaten festgelegt. Die foreach-Schleife durchläuft dieses Array komplett und weist bei jedem Durchlauf der Variablen $z eine der Zutaten aus dem Array zu. Diese Variable wird in der Ausgabe mit echo in die checkbox-Eingabefelder geschrieben.

Der Vorteil bei dieser Methode im Vergleich zur HTML-Variante ist vielleicht auf den ersten Blick nicht erkennbar. Stellen Sie sich allerdings vor, Sie hätten nicht nur fünf Zutaten zu verwalten, sondern einige hundert sollen zur Auswahl stehen, und diese Auswahl soll möglicherweise auch noch häufig verändert werden. Bei solch großen Mengen ist die Verwendung von Arrays deutlich einfacher als die Pflege direkt in HTML.

1.5.7 Bestellung als E-Mail absenden

Die eigentliche Bestellung wird bisher nur als Text wieder im Browser des Bestellers dargestellt. Der Pizzabäcker erfährt bislang gar nicht, dass ein Interessent diese Pizza bestellen möchte. Im einfachsten Fall soll die Bestellung jetzt an den Pizzabäcker per E-Mail versendet werden. Dazu benötigen Sie die E-Mail Adresse des Pizzabäckers. Da bei der Standardinstallation auf jedem SuSE Linux *sendmail* (oder ein anderer MTA) installiert ist, kann der User pizza auf jeden Fall mit der E-Mail Adresse pizza@localhost erreicht werden. Sie sollten allerdings (hauptsächlich aus optischen Gründen) noch einen Alias in /etc/aliases wie folgt anlegen. Bearbeiten Sie als Benutzer root die Datei /etc/aliases mit einem beliebigem Texteditor und fügen Sie die folgende Zeile am Ende dieser Datei an:

```
web.pizza:              pizza
```

Um diesen Alias zu aktivieren, geben Sie nun noch (wieder als `root`) folgendes Kommando in einer Konsole ein:

```
linux: # newaliases
```

Falls Sie die E-Mail an einen anderen Rechner versenden möchten, so müssen Sie *sendmail* zuvor für den Einsatz mit Remote-Rechnern konfigurieren. Dies würde allerdings den Umfang dieses Abschnitts sprengen, daher nur noch kurz der Hinweis, wie Sie `sendmail` starten können (nur notwendig bis SuSE Linux 7.3, seit Version 8.0 wird *sendmail* in den Runlevels 3 & 5 automatisch gestartet):

```
linux: # rcsendmail start
```

Jetzt kann die Datei `bestellungen-2.php` bearbeitet werden, um die Bestellung als E-Mail an den Benutzer `pizza` abzuschicken. Bearbeiten Sie diese Datei, sodass sie folgenden Inhalt hat:

```
<html>
<head><title>Pizzabäcker Bestellservice</title></head>
<body>
<h1 align="center">Vielen Dank</h1>
Folgenden Daten wurden vom Formular übermittelt:<br>
<?= $anrede; ?><br>
<?php
echo "$vorname $name<br>\n";
echo "$strasse<br>\n";
echo "$plz $ort<br>\n";
echo "$email<br>\n";
?>
Sie möchten eine Pizza mit folgenden Zutaten:
<?php
$bestellung="$anrede\n$vorname $name\n$strasse\n$plz $ort\n";
$bestellung.="$email\n\n";
$bestellung.="bestellt eine Pizza mit folgenden Zutaten:\n\n";

# $zutaten ist ein Array, daher muss
# eine Schleife die Zutaten ausgeben:
foreach($zutaten as $z){
        echo "$z ";
        $bestellung.="$z\n";
}
if($email){
        $header="From: $vorname $name <$email>\n";
}else{
        $header="From: Web Pizza <web.pizza@localhost>\n";
}
```

```
mail("web.pizza@localhost","Pizzabestellung",
        $bestellung,$header);
?>
</body>
</html>
```

Dieses PHP-Skript erzeugt für den Besteller wieder die gleiche Ergebnisseite, wie sie auch schon in der ersten Version angezeigt wurde. Allerdings wird „nebenbei“ die Bestellung noch als E-Mail an die Adresse `web.pizza` geschickt. Hierfür wird zuerst in die Variable `$bestellung` der Inhalt der Variablen (`$anrede, $vorname, $name, $strasse, $ort, $email`), die vom Benutzer im Bestellformular eingetragen wurden, geschrieben. An diese Benutzerdaten wird der Text `bestellt eine Pizza mit folgenden Zutaten:` angehängt. Dazu wird der Zuweisungsoperator (`.=`) verwendet (siehe Abschnitt 5.6.2.3).

Mit Hilfe der `foreach`-Schleife wird schließlich an den Inhalt dieser Variablen nun noch jede Zutat, die gewählt wurde, angehängt.

Im letzten Teil wird mit der Kontrollstruktur `if` (siehe Abschnitt 5.7.1) zuerst überprüft, ob die Variable `$email` einen Inhalt hat. Falls dies der Fall ist, so wird in die Variable `$header` als Absender der Pizzabesteller und dessen E-Mail-Adresse eingetragen. Falls keine E-Mail Adresse übermittelt wurde, wird der oben angelegte Alias als Absender festgelegt (dieser Alias wird nur deshalb verwendet, da *sendmail* den Header umschreibt, falls kein Punkt enthalten ist).

Zu guter Letzt wird die E-Mail an den Empfänger `web.pizza@localhost` mit dem Betreff `Pizzabestellung`, dem Inhalt `$bestellung` und dem zusätzlichen Header `$header` (für den Absender) abgeschickt. Eine genauere Erläuterung der `mail`-Funktion finden Sie im Abschnitt 5.14.

1.5.8 Zutaten aus einer Textdatei

Jetzt können Sie sich wieder den Anpassungen der Webseite des Pizzabäckers widmen. Momentan werden die Zutaten nach wie vor noch statisch in der Datei `bestellung.php` festgelegt. Ihr Pizzabäcker muss diese Daten allerdings von Zeit zu Zeit anpassen, um hier auf saisonale Gegebenheiten reagieren zu können. Aus diesem Grund sollen die Zutaten in einer externen Datei gespeichert werden. Diese Datei soll jede der Zutaten jeweils in einer eigenen Zeile enthalten. Legen Sie daher als Benutzer `pizza` mit einem Texteditor die Datei `zutaten.txt` im Home-Verzeichnis des Users `pizza` an. Tragen Sie folgende Zutaten ein:

```
Salami
Schinken
Champignons
```

```
Peperoni
Paprika
Zwiebeln
```

Diese Datei soll nun von der Seite `bestellung.php` ausgewertet werden. Die Zutaten sollen nicht mehr aus dem statischen Array kommen, sondern dieses Array wird nun aus der Textdatei erzeugt. Öffnen Sie die Datei `bestellung.php` und kommentieren Sie die statische Arrayzuweisung aus, indem Sie dieser Zeile das Doppelkreuz voranstellen bzw. die Zeilen mit `/*` und `*/` einklammern (siehe Abschnitt 5.3). Alternativ können Sie die komplette Arrayzuweisung auch aus dieser Seite entfernen.

Jetzt muss dieses Array `$zutaten` aus dem Inhalt der Datei `zutaten.txt` generiert werden. Ergänzen Sie dazu den nachfolgenden Code in Ihrem Skript:

```
⋮
<?php
$fh=fopen("../zutaten.txt","r");
while($zeile=fgets($fh,4096)){
        $zutaten[]=$zeile;
}
fclose($fh);
/* $zutaten=array("Salami",
        "Schinken",
        "Champignons",
        "Peperoni",
        "Paprika"); */
foreach($zutaten as $z){
        echo "<input type=\"checkbox\" name=\"zutaten[]\"
              value=\"$z\">$z<br>\n";
}
?>
⋮
```

Der Befehl `fopen` öffnet die Datei `zutaten.txt` im übergeordneten Verzeichnis zum Lesen (dies bewirkt der zweite Parameter `r`). Der Rückgabewert wird zur weiteren Verwendung in der Variable `$fh` abgespeichert (dies ist der so genannte „Dateihandle“). Im Anschluss wird eine `while`-Schleife solange durchlaufen, wie Sie aus der Datei `zutaten.txt` noch Zeilen auslesen (`fgets`). Jede dieser Zeilen wird in der Variablen `$zeile` zwischengespeichert. Bei jedem Schleifendurchlauf wird der Inhalt der temporären Variablen `$zeile` an das Array `$zutaten` angehängt (siehe Abschnitt 5.6.4). Das letzte Kommando `fclose` schließt den Dateihandle, der in der Variablen `$fh` gespeichert ist, wieder. Weitere Informationen zum Umgang mit Dateien finden Sie im Abschnitt 5.11.

Nun kann Ihr Pizzabäcker durch einfaches Bearbeiten der Datei `zutaten.txt` seine Zutatenliste verändern. Wenn Sie die Seite `bestellen.php` im Browserfenster aktualisieren, müssten Sie bereits eine Veränderung auf der Bestellseite feststellen. In der Textdatei sind als Zutat noch `Zwiebeln` hinzugekommen. Dieser Eintrag muss auf Ihrer Bestellseite nun ebenfalls als Checkbox sichtbar sein.

1.5.9 Einen geschützten Bereich festlegen

Die Textdatei mit den Zutaten ist ja bereits eine Erleichterung für den Pizzabäcker, um die Zuaten unabhängig von HTML und PHP eingeben zu können. Allerdings ist dafür nach wie vor ein direkter Zugriff auf den Linux-Rechner durch den Pizzabäcker notwendig. Entweder muss der Pizzabäcker sich auf dem Rechner direkt anmelden – per Telnet oder SSH – oder die Datei `zutaten.txt` muss mit Hilfe von FTP auf den Server geladen werden.

Im Idealfall sollte der Pizzabäcker aber auch über eine Webschnittstelle in der Lage sein, die Zutaten zu bearbeiten. Diese Webschnittstelle soll es ermöglichen, bereits existierende Zutaten zu bearbeiten, einzelne Zutaten zu löschen und neue Zutaten hinzuzufügen.

Da allerdings diese Bearbeitung der Zutaten nur vom Pizzabäcker selbst erledigt werden soll, muss hierfür ein eigener administrativer Bereich erstellt werden. Im einfachsten Fall sollte sich der Pizzabäcker auf seiner Administrationsseite per Benutzername und Passwort authentifizieren und dann mit der Bearbeitung der Zutaten beginnen können.

Apache bietet hierfür eine Möglichkeit, bestimmte Verzeichnisse über einen Benutzernamen und ein Passwort zu schützen. Seit SuSE Linux 7.3 ist dieses Feature nun standardmäßig aktiviert, sodass Sie als Benutzer `pizza` einen geschützten Bereich sofort anlegen können. Achtung! Bei einigen älteren Versionen muss dieses Feature erst aktiviert werden! Hierfür müssten Sie der Direktive `AllowOverride` für die benutzereigenen Webverzeichnisse `public_html` den Wert `AuthConfig` zuweisen. Falls das bei Ihnen der Fall sein sollte, dann sollten Sie Abschnitt 3.3.9 für weitere Informationen zu Rate ziehen.

Legen Sie zuerst ein Verzeichnis an, in dem alle administrativen Webseiten abgelegt werden sollen. Da dieses Verzeichnis vom Web aus erreichbar sein soll, muss es unterhalb von `~/public_html` angelegt werden. Nennen Sie das Verzeichnis `admin`. Dies kann natürlich sowohl mit dem *Konqueror* als auch über die Konsole durchgeführt werden.

Das folgende Beispiel zeigt die Konsolenvariante:

```
pizza@linux:~ >  cd public_html
pizza@linux:~/public_html >  mkdir admin
pizza@linux:~/public_html >  cd admin
```

Für einen Bereich, der mit Benutzername und Passwort geschützt sein soll, müssen Sie eine Datei anlegen, in der die Benutzernamen und die zugehörigen Passwörter abgelegt sind. Mit Apache ist hier das Tool `htpasswd` mitgeliefert, das diese Aufgabe für Sie erledigt (siehe Abschnitt 3.7.2). Geben Sie in einer Konsole folgendes Kommando ein, um eine neue Benutzer/Passwortdatei anzulegen:

```
pizza@linux:~/public_html/admin >  htpasswd -c ~/.htpasswd \
> web-admin
New password:
Re-type new password:
Adding password for user web-admin
```

Durch den Parameter `-c` legen Sie fest, dass Sie eine neue Datei anlegen möchten (für weitere Einträge in der Datei ist dieser Parameter nicht mehr anzugeben!). Die Datei, die angelegt wird, soll im Home-Verzeichnis des Pizzabäckers angelegt werden und `.htpasswd` heißen. Die letzte Angabe bei diesem Kommando (`web-admin`) legt fest, welcher Benutzername angelegt werden soll. Im Anschluss daran werden Sie nach dem Passwort für diesen Benutzer gefragt. Verwenden Sie hier keinen Benutzer, der im System vorhanden ist und auch kein Passwort, dass Sie im System verwenden!

Anschließend muss noch angegeben werden, dass alle Seiten, die im Verzeichnis `admin` und den darunter befindlichen Verzeichnissen sind, durch Benutzername und Passwort geschützt werden sollen. Diese Funktion erledigt die Datei `.htaccess`. Legen Sie im Verzeichnis `admin` die Datei `.htaccess` mit folgendem Inhalt an:

```
AuthName "Administration der Pizza"
AuthType Basic
AuthUserFile /home/pizza/.htpasswd
require user web-admin
```

Die Angabe `AuthName` gibt den Namen des Authentifizierungsbereichs an. Der Browser wird allen Seiten, die den gleichen Authentifizierungsbereich verwenden, den Benutzernamen und das Passwort senden, das Sie bei Ihrer ersten Anmeldung dieser Sitzung eingegeben haben. Auf diese Weise muss der Benutzer die Authentifizierung nur einmal durchführen und kann ab dann mit den administrativen Webseiten ohne weitere Anmeldungen arbeiten – zumindest bis der Browser beendet wird.

Bei `AuthUserFile` geben Sie die Datei an, die die Kombinationen aus Benutzername und Passwort enthält, die zur Anmeldung verwendet werden dürfen.

Die letzte Angabe `require` legt noch fest, welcher Benutzer für diesen Bereich erforderlich ist. Dies ist wichtig, da in Ihrer Passwortdatei ja mehrere Benutzer eingetragen sein können, aber für diesen administrativen Bereich soll nur der Benutzer `web-admin` zugelassen sein.

Versuchen Sie nun, in Ihrem Web-Browser einen Zugriff auf den administrativen Bereich unter dem URL `http://web-pizza.local/admin/` durchzuführen. Sie erhalten die Aufforderung, sich mit Benutzername und Passwort anzumelden. Falls Sie als Benutzername `web-admin` und als Passwort Ihr oben festgelegtes Passwort eingeben, so zeigt Ihnen der Browser das leere `admin` Verzeichnis an.

Weitere Informationen zu `htpasswd` und `.htaccess` finden Sie im Abschnitt 3.3.9.

1.5.10 Zutaten im geschützten Bereich bearbeiten

Um in Ihrem geschützten Bereich nun die Zutaten zu bearbeiten, müssen Sie noch einige Dateien anlegen. Die erste Datei soll die Datei `index.html` sein, in der eine Auswahl der möglichen Aktionen präsentiert wird. Legen Sie im Verzeichnis `admin` die Datei `index.html` mit folgendem Inhalt an:

```
<html>
<head><title>Pizzaadministration</title></head>
<body>
<h3>Pizzaadministration</h3>
Wählen Sie eine Funktion:<br>
<ul>
<li><a href="loeschen.php">Zutaten löschen</a></li>
<li><a href="neu.php">Neue Zutat hinzufügen</a></li>
</ul>
</body>
</html>
```

Von nun an kann Ihr Pizzabäcker im administrativen Bereich wählen, ob er Zutaten aus der Liste entfernen oder neue Zutaten hinzufügen möchte.

Betrachten Sie zunächst einmal die Funktion „Löschen“. Hierfür legen Sie eine weitere Datei mit dem Namen `loeschen.php` an. Diese Datei sollte folgenden Inhalts sein:

```
<html>
<head><title>Pizza Zutaten löschen</title></head>
<body>
<h3>Zutaten löschen</h3>
Zutaten werden gelöscht, falls sie nicht angehakt sind.<br>
<form action="speichern.php">
<?php
$fh=fopen("../../zutaten.txt","r");
while($zeile=fgets($fh,4096)){
        $zutaten[]=$zeile;
```

```
}
fclose($fh);
foreach($zutaten as $z){
        echo "<input type=\"checkbox\" name=\"zutaten[]\"
              value=\"$z\" CHECKED>$z<br>\n";
}
?>
<input type="submit" value="speichern">
</form>
</body>
</html>
```

In dieser Datei wird die komplette Zutatenliste ausgegeben, wie es bereits bei der Pizzabestellungsseite der Fall war. Der einzige große Unterschied besteht darin, dass in der `checkbox` noch der Parameter `CHECKED` vorhanden ist. Die Standardeinstellung bei den Zutaten soll ja bewirken, dass die Zutaten beibehalten werden. Nur diejenigen Zutaten, die der Pizzabäcker entfernt haben möchte, sollen durch explizites Wegklicken aus der Zutatenliste gelöscht werden.

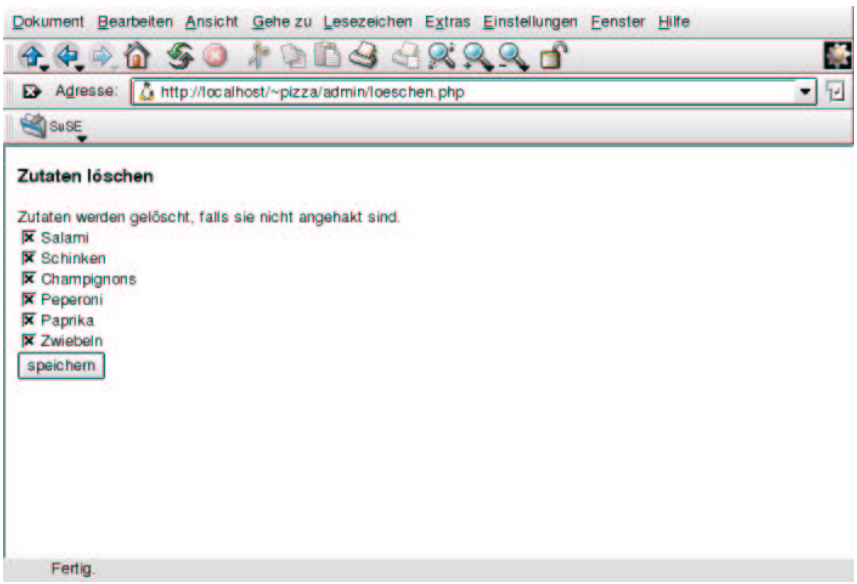

Abbildung 1.3: Auswahl der Zutaten zum Löschen

`speichern.php` ist die nächste Datei, die Sie für diese Funktion benötigen. Legen Sie auch diese Datei im Verzeichnis `admin` an und fügen Sie folgenden Inhalt ein:

```
<html>
<head><title>Pizza Zutaten speichern</title></head>
<body>
<h3>Zutaten gespeichert</h3>
Folgende Zutaten wurden gespeichert:<br>
<?php
$fh=fopen("../../zutaten.txt","w");
foreach($zutaten as $z){
        echo "$z<br>\n";
        fwrite($fh, "$z\n");
}
fclose($fh);
?>
</body>
</html>
```

Hier wird nun die Datei `zutaten.txt` zum Schreiben geöffnet. Im aktuellen Fall sogar zum Überschreiben. Durch die Angabe von `w` wird die Datei auf jeden Fall neu angelegt. Das heißt, dass die alten gespeicherten Zutaten durch die Auswahl des Pizzabäckers ersetzt werden. Hierfür ist es allerdings erforderlich, dass der Benutzer, der den PHP-Prozess ausführt, auch Schreibberechtigung auf dieser Datei hat. Daher müssen Sie die Berechtigungen für diese Datei für `Sonstige` auf jeden Fall auf `rw` setzen. Dies kann wieder durch Eingaben in der Konsole oder mit dem *Konqueror* geschehen.

Hier der Konsolenbefehl:

```
pizza@linux:~ >  chmod o+w ~/zutaten.txt
```

Mit der Funktion `fwrite` schreibt die obige Routine die einzelnen Zutaten in die Textdatei. Achten Sie dabei darauf, dass nach jeder Zutat auch eine neue Zeile begonnen werden muss! Dies erledigen Sie durch Anhängen von `\n`.

Die Funktion für eine neue Zutat sieht hierbei noch etwas einfacher aus. Dazu werden ebenfalls zwei Dateien benötigt. Die erste Datei stellt nur ein Formular dar, in dem der Benutzer den Namen der neuen Zutat eingeben kann. Speichern Sie dieses Formular unter dem Namen `neu.php`:

```
<html>
<head><title>Neue Pizzazutat</title></head>
<body>
<h3>Neue Zutat</h3>
<form action="speichern-neu.php">
Zutat: <input type="text" name="zutat"><br>
<input type="submit">
</body>
</html>
```

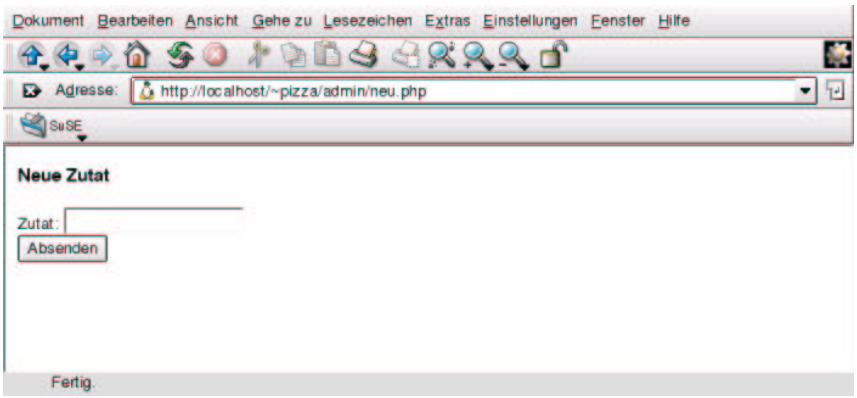

Abbildung 1.4: Eintragen einer neuen Zutat

Das eigentliche Speichern der neuen Zutat übernimmt dann die folgende Datei, die Sie unter dem Namen `speichern-neu.php` ablegen:

```
<html>
<head><title>Neue Pizza Zutat speichern</title></head>
<body>
<h3>Zutat gespeichert</h3>
Folgende Zutat wurde gespeichert:<br>
<?php
$fh=fopen("../../zutaten.txt","a");
echo "$zutat<br>\n";
fwrite($fh, "$zutat\n");
fclose($fh);
?>
</body>
</html>
```

Diese Datei entspricht in weiten Teilen der Datei, die Sie bereits beim Löschen der Zutaten gesehen haben. Der wichtigste Unterschied besteht nun darin, wie die Datei zum Schreiben geöffnet wird. Sie geben hier als Modus `a` an. Dies bewirkt, dass neue Informationen, die geschrieben werden sollen, an die Datei angehängt werden. Dadurch bleiben die alten Informationen in Ihrer Zutatenliste jeweils erhalten und die neue Zutat wird ans Ende der Datei angehängt.

1.5.11 Eine Datenbank für den Pizzabäcker

Die bisherige Zutatenverwaltung soll nun auf ein Datenbanksystem umgestellt werden. Um Werte in eine Datenbank als Pizzabäcker schreiben zu können, müs-

sen Sie allerdings zuerst einmal diese Datenbank anlegen und für den Benutzer `pizza` freigeben.

Das Anlegen und Verwalten von Datenbanken kann mit dem textbasierenden Tool `mysqladmin` (dies wird im Abschnitt 4.11.1 näher erläutert) oder auch mit `phpMyAdmin`, das im Kapitel 6 ausführlicher vorgestellt wird, erledigt werden. Diese Aufgabe muss vom Datenbankadministrator durchgeführt werden. Als Standard ist das der Datenbankbenutzer `root`. Bevor Sie mit `phpMyAdmin` arbeiten können, muss in der Konfigurationsdatei von `phpMyAdmin` zuerst der Benutzername und das Passwort des Datenbankadministrators eingetragen werden (oder bei Versionen >= 2.2.3 als Authentifizierung `cookie` festgelegt werden).

Hier die Variante mit `mysqladmin` (Sie können dieses Kommando als Benutzer `pizza` ausführen, da Sie ja das Passwort des Datenbankadministrators `root` kennen):

```
pizza@linux:~ >  mysqladmin create pizza -u root -p
Enter password:
```

Das Passwort, nach dem Sie gefragt werden, ist jenes, welches Sie nach dem ersten Start von MySQL ebenfalls mit `mysqladmin` festgelegt haben (siehe Abschnitt 1.5.1).

Jetzt muss dem Datenbankbenutzer `pizza` noch die Berechtigung gegeben werden, um auf die Datenbank `pizza` zugreifen zu können. Dies wird ebenfalls mit `phpMyAdmin` oder aber mit dem Textclient `mysql` erledigt. Weitere Informationen zum Eintragen der Berechtigungen für Benutzer in MySQL finden Sie im Abschnitt 4.12.2. Die Bedienung des textbasierenden Tools `mysql` können Sie im Abschnitt 4.7 nachlesen.

Geben Sie in einer Konsole folgendes Kommando ein:

```
pizza@linux:~ >  mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22 to server version: 3.23.48-Max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,INDEX,ALTER,CREATE,DROP
    -> ON pizza.* TO pizza@localhost IDENTIFIED BY 'passwort'
    -> WITH GRANT OPTION;
Query OK, 0 rows affected (0.01 sec)

mysql> quit
Bye
```

Tragen Sie statt `passwort` das Passwort ein, das für den Zugriff auf die Datenbank `pizza` vom Benutzer `pizza` aus verwendet werden soll. Um den Zugriff als Benutzer `pizza` auf die Datenbank `pizza` zu testen, geben Sie folgendes Kommando ein:

```
pizza@linux:~ >  mysql -u pizza -p
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 27 to server version: 3.23.41-Max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use pizza
Database changed
mysql> quit
Bye
```

Damit ist gewährleistet, dass Sie als Benutzer `pizza` bereits jetzt Zugriff auf die eben erstellte Datenbank haben.

1.5.11.1 Eine Tabelle für die Zutaten

Für die Tabelle, in der die Zutaten gespeichert werden, sollten Sie zunächst noch einige Vorüberlegungen treffen. Zunächst müssen Sie überlegen, welche Informationen noch zu jeder Zutat gespeichert werden sollen. Hier liegt es nahe, dem Pizzabäcker die Arbeit etwas zu erleichtern und die Zutaten nicht immer gleich pauschal aus der Zutatenliste zu entfernen, sondern die Zutaten nur zu deaktivieren bzw. später auch wieder zu reaktivieren. Daher soll in der Tabelle für die Zutaten eine Spalte vorhanden sein, die angibt, ob diese Zutat derzeit verfügbar ist oder nicht. Dementsprechend soll dann der normale Benutzer diese Zutat sehen oder eben nicht.

Die weiteren Vorüberlegungen betreffen die Art und Weise, wie die Informationen in der Tabelle abgelegt werden sollen. Diese verschiedenen Speichermöglichkeiten zeigt Ihnen Abschnitt 4.2 genauer.

Legen Sie eine Tabelle mit Namen `zutaten` an, in der die beiden folgenden Felder definiert sind:

Feldname	Felddefinition
`Zutat`	`VARCHAR(30) NOT NULL PRIMARY KEY`
`Verfuegbar`	`ENUM('j','n') NOT NULL`

Auch hier haben Sie wieder die Möglichkeit, diese Tabelle mit `phpMyAdmin` zu erstellen, oder aber mit dem Kommandozeilentool `mysql`.

Hier das Beispiel für die Erstellung mit `mysql`:

```
pizza@linux:~ >  mysql -u pizza -p
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29 to server version: 3.23.48-Max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use pizza
Database changed
mysql> CREATE TABLE zutaten (
    -> Zutat VARCHAR(30) NOT NULL PRIMARY KEY,
    -> Verfuegbar ENUM('j','n') NOT NULL);
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

Zunächst teilen Sie dem SQL-Server mit, dass die nachfolgenden Kommandos auf die Datenbank `pizza` angewendet werden sollen (`use pizza`). Im anschließenden SQL-Befehl `CREATE TABLE` legen Sie die Tabelle `zutaten` mit den obigen Feldern und den zugehörigen Felddefinitionen an. Nähere Informationen zu `CREATE TABLE` stehen im Anhang C.1.

1.5.11.2 Die Zutatentabelle mit Werten füttern

Da die Tabelle `zutaten` derzeit noch leer ist, sollten Sie zunächst einmal einige Einträge vornehmen. Auch dies kann wieder auf verschiedene Arten geschehen: Natürlich bietet sich auf der einen Seite wieder `phpMyAdmin` an oder alternativ wiederum das textbasierende Tool `mysql`. Beim Eintragen dieser Werte wird das Feld `Verfuegbar` unten nicht verwendet. MySQL trägt in diesem Fall den Standardwert (`j`) (der erste eingetragene Wert des `ENUM`-Feldes) ein. Dadurch werden vorerst alle Zutaten als verfügbar gekennzeichnet.

Hier das Beispiel mit `mysql`:

```
pizza@linux: > mysql -u pizza -p
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29 to server version: 3.23.48-Max-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use pizza
Database changed
mysql> INSERT INTO zutaten (Zutat) VALUES ('Salami');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO zutaten (Zutat) VALUES ('Schinken');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO zutaten (Zutat) VALUES ('Champignons');
Query OK, 1 row affected (0.00 sec)

mysql> quit
Bye
```

Durch das SQL-Kommando `INSERT INTO` können Sie Werte in Tabellen eintragen. Als ersten Parameter geben Sie immer die Tabelle an, in die die Werte aufgenommen werden sollen. Im Anschluss an den Tabellennamen steht in runden Klammern eine Liste mit Feldnamen, in die Sie einen Wert eintragen möchten. Nach der schließenden runden Klammer steht das Schlüsselwort `VALUES`, um anzuzeigen, dass in der nächsten runden Klammer die zugehörigen Werte eingetragen werden. Achten Sie beim Eintragen von Werten darauf, dass diese in Anführungszeichen geschrieben werden, falls es sich nicht um Zahlenwerte handelt (auch um ein Datum müssen Anführungszeichen gesetzt werden!).

Eine genauere Erläuterung zum `INSERT INTO`-Kommando bekommen Sie im Abschnitt 4.8.6.

1.5.11.3 Zutaten aus der Tabelle anzeigen

Da Sie nun die Zutaten in einer Datenbank gespeichert haben, sollen diese Zutaten ab sofort auch in Ihrer Bestellseite sichtbar sein. Derzeit wird das Zutatenarray aus einer Datei generiert. Dieser Teil muss nun durch einen Datenbankzugriff ersetzt werden. Sie können den Teil wieder durch Kommentare einfach deaktivieren oder aber auch ganz aus Ihrem Quellcode löschen. Führen Sie dies von `$fh=fopen("../zutaten.txt",r");` bis zu `fclose($fh);` durch und ersetzen Sie diesen Teil wie folgt:

```
$dbh=mysql_connect("localhost","pizza","passwort");
mysql_select_db("pizza");
$result=mysql_query("SELECT * FROM zutaten");
while($row=mysql_fetch_array($result)){
        if($row["Verfuegbar"]=="j"){
                $zutaten[]=$row["Zutat"];
        }
}
mysql_close($dbh);
```

In der ersten Zeile dieses veränderten Codes bauen Sie mit dem Kommando `mysql_connect` eine Verbindung mit dem Datenbankserver auf. Der Datenbankserver ist in diesem Fall der eigene Rechner, daher können (bzw. sogar müs-

sen!) Sie hier als Rechner `localhost` angeben. Der nächste Parameter ist der Datenbankbenutzername, mit dem auf die Datenbank zugegriffen werden soll. Sie haben oben mit dem `GRANT`-Kommando genau diesen Benutzer für den Zugriff von `localhost` angelegt. Die letzte Angabe ist hier das Passwort, dass Sie ebenfalls oben mit dem `GRANT`-Kommando in MySQL hinterlegt haben. Der Rückgabewert dieser Funktion ist der Datenbankhandle, mit dem die weitere Kommunikation mit dem Server durchgeführt wird. Dieser Handle wird daher in einer Variablen `$dbh` abgelegt. Weitere Informationen zu `mysql_connect` finden Sie im Abschnitt 5.12.1.

Das Kommando `mysql_select_db` legt fest, mit welcher Datenbank Sie auf dem Server arbeiten möchten. Durch das `GRANT` Kommando weiter oben haben Sie festgelegt, dass der Datenbankbenutzer `pizza` nur Zugriff auf die Datenbank `pizza` hat. In dieser Datenbank ist auch die Tabelle `zutaten` abgelegt, daher wählen Sie hier `pizza` aus. Weitere Infos hierzu stehen im Abschnitt 5.12.4.

Die eigentliche Abfrage wird mit dem Kommando `mysql_query` an den Datenbankserver abgeschickt. Die Abfrage, die Sie hier an den Server senden, kann jedes beliebige SQL-Kommando enthalten. Im aktuellen Beispiel wird als Abfrage das `SELECT`-Kommando (siehe Abschnitt 4.8.7) abgeschickt, um sämtliche Informationen (`*`) ohne Bedingungen (keine `WHERE`-Klausel) aus der Tabelle `zutaten` zu holen. Ein Zeiger auf diese Ergebnismenge wird in der Variablen `$result` abgelegt.

Da die Ergebnismenge (die Zutaten) normalerweise nicht nur aus einem Element besteht, muss mit Hilfe einer `while`-Schleife jedes dieser Ergebnisse ausgelesen werden, wobei das Auslesen selbst die Funktion `mysql_fetch_array` erledigt. Diese Funktion liefert einen Hash (siehe Abschnitt 5.6.4) als Rückgabewert. Dieser Hash hat als Schlüssel den jeweiligen Feldnamen aus der SQL-Abfrage und als Wert den eigentlichen Feldinhalt. Informationen über die Funktion `mysql_fetch_array` stehen im Abschnitt 5.12.9.

Da Sie ja in der Tabelle bereits vorbereitet haben, dass nur Zutaten angezeigt werden sollen, die momentan verfügbar sind, wird in der nächsten Zeile geprüft, ob die aktuelle Zutat auch bei `Verfügbar` den Wert `j` stehen hat. Falls dies der Fall ist, so wird das Array `$zutaten` mit der aktuell aus der Datenbank ausgelesenen Zutat `$row["Zutat"]` ergänzt.

Am Ende der Routine wird die Verbindung mit der Datenbank über den Befehl `mysql_close` wieder geschlossen.

Zum Ausprobieren, ob die Zutaten auch tatsächlich aus der Datenbank gelesen werden, rufen Sie den URL `http://web-pizza.local/bestellen.php` auf. Hier sollten im Moment nur die drei Zutaten („Salami“, „Schinken“, „Champignons“) angezeigt werden, die Sie mit den `INSERT`-Kommandos oben eingetragen haben.

1.5.11.4 Neue Zutaten in die Tabelle schreiben

Der etwas einfachere Teil der Administration für die Zutaten betrifft das Aufnehmen einer neuen Zutat in die Tabelle. Hierfür könnnen Sie die Administrationsseiten `index.html` und `neu.php` unverändert lassen. Die einzige Datei, die geändert werden muss, ist die Datei `speichern-neu.php`. Ändern Sie diese Datei so ab, dass Folgendes darin enthalten ist:

```
<html>
<head><title>Neue Pizza-Zutat speichern</title></head>
<body>
<h3>Zutat gespeichert</h3>
Folgende Zutat wurde gespeichert:<br>
<?php
$dbh=mysql_connect("localhost","pizza","passwort");
mysql_select_db("pizza");
mysql_query("INSERT INTO zutaten (Zutat) VALUES ('$zutat')");
echo "$zutat<br>\n";
mysql_close($dbh);
?>
<a href="index.html">Startseite der Administration</a>
</body>
</html>
```

Hier wird wieder zuerst eine Verbindung mit dem Datenbankserver über den Befehl `mysql_connect` aufgebaut. Im Anschluss daran wählen Sie erneut die Datenbank `pizza` aus, um die nachfolgende(n) Aktion(en) mit dieser Datenbank auszuführen.

Der interessante Teil ist hier das Kommando `mysql_query`, mit dem Sie den SQL-Befehl an den Datenbankserver schicken. Da in diesem Fall kein Ergebnis vom Datenbankserver benötigt wird, wird der Rückgabewert dieser Funktion nicht in einer Variablen abgelegt. Das SQL-Kommando zum Speichern von Werten in der Tabelle `zutaten` ist wieder das `INSERT INTO`-Kommando, das Sie bereits oben gesehen haben. Hier wird als Wert der Inhalt der Variablen `$zutat`, die im Formular eingetragen wurde, in die Tabelle gespeichert. Zusätzlich wird in der nächsten Zeile noch für eine Bildschirmausgabe gesorgt.

Da die `Zutat` in der Tabelle als `PRIMARY KEY` festgelegt worden ist, können Sie auf diese Weise auch keine Zutat doppelt in die Tabelle speichern. Primärschlüssel haben immer als Eigenschaft, dass sie eindeutig sein müssen. Weitere Informationen über Primärschlüssel sind im Abschnitt 4.3 nachzulesen.

1.5.11.5 Zutaten aktivieren/deaktivieren

Nach der relativ einfachen Funktion, mit der Sie eine neue Zutat hinzufügen können, soll nun noch die etwas komplexere Variante betrachtet werden, mit der Sie bestehende Zutaten aktivieren bzw. deaktivieren können. Diese Funktion soll es dem Pizzabäcker ermöglichen, Zutaten nur einmal in der Datenbank aufzunehmen und dann die Zutaten je nach Bedarf anzeigen zu lassen oder aus der Liste herauszulassen. Idealerweise sollte in der Auswahl auch gleich angezeigt werden, welche Artikel derzeit in der Tabelle als verfügbar markiert sind.

Legen Sie die folgende Datei in Ihrem Administrationsverzeichnis `admin` mit dem Dateinamen `aendern.php` an.

```
<html>
<body>
<form action="aendern-db.php">
<table>
<tr>
<th>Zutat</th><th>Verfügbar?</th>
<?php
$dbh=mysql_connect("localhost","pizza","passwort");
mysql_select_db("pizza");
$result=mysql_query("SELECT * FROM zutaten ORDER BY Zutat");
while($row=mysql_fetch_array($result)){
        if($row["Verfuegbar"]=="j"){
                $checkj="CHECKED";
                $checkn="";
        }else{
                $checkn="CHECKED";
                $checkj="";
        }
        echo "<tr>\n";
        echo "<td>".$row['Zutat']."</td>";
        echo "<td><input type=\"radio\"
            name=\"Verfuegbar[$row[Zutat]]\"
            $checkj
            value=\"j\">j\n";
        echo "<input type=\"radio\"
            name=\"Verfuegbar[$row[Zutat]]\"
            $checkn
            value=\"n\">n</td>\n";
        echo "</tr>\n";
}
mysql_close($dbh);
?>
</table>
```

```
<input type="submit" value="speichern">
</form>
<a href="index.html">Startseite der Administration</a>
</body>
</html>
```

Hier wurde in der Abfrage der Zutaten am `SELECT`-Befehl noch etwas ergänzt: Durch die Angabe von `ORDER BY Zutat` legen Sie fest, dass die Werte, die aus der Datenbank abgefragt werden, nach der `Zutat` alphabetisch sortiert werden sollen. Dies soll dem Pizzabäcker das Durchsuchen der einzelnen Zutaten erleichtern. Den Zusatz könnten Sie übrigens auch auf Ihrer Bestellseite bei `SELECT` ergänzen, um auch dort eine sortierte Ausgabe der Zutaten zu bekommen. Weitere Informationen zu `ORDER BY` finden Sie im Abschnitt 4.8.9.

Der Test `if($row["Verfuegbar"])=="j")` überprüft, ob die gerade gewählte Zutat als verfügbar in der Tabelle markiert ist. Die beiden Variablen `$checkj` und `$checkn` werden je nachdem, ob die Zutat verfügbar ist oder nicht, auf `CHECKED` und auf leer gesetzt. Der Text `CHECKED` wird als Parameter bei den Radiobuttons hinzugefügt, um den Radiobutton zu aktivieren, der für die aktuelle Zutat den korrekten Zustand enthält.

Bei den Radiobuttons wird als Name `Verfuegbar[Zutat]` festgelegt. Wobei `Zutat` durch die aktuelle Zutat ersetzt wird. Beispielsweise steht bei der Zutat `Salami` als Name `Verfuegbar[Salami]`. Als Wert des jeweiligen Radiobuttons wird entweder `j` oder eben `n` angegeben. Und je nachdem, ob diese Zutat eben verfügbar war oder nicht, wird noch der Parameter `CHECKED` gesetzt. Dies erzeugt auf der PHP-Seite, die später für die Auswertung zuständig ist, einen Hash, der jede Zutat als Schlüssel enthält und dessen zugehöriger Wert jeweils `j` oder `n` ist. Dies erzeugt ein Formular für den Administrator dieser Seite, in dem er für jede einzelne Zutat angeben kann, ob diese Zutat verfügbar ist oder nicht.

Testen Sie dieses Formular unter dem URL:
`http://web-pizza.local/admin/aendern.php`

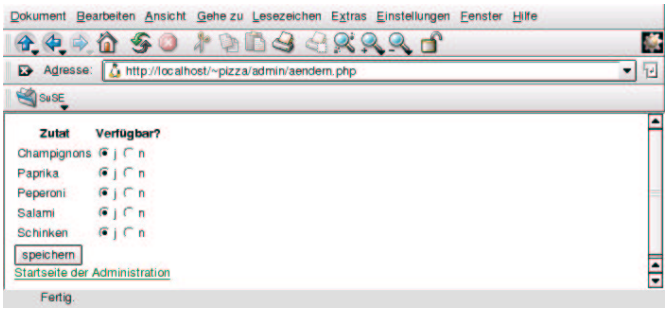

Abbildung 1.5: Auswahl der aktiven/inaktiven Zutaten

Nun benötigen Sie noch die Seite `aendern-db.php`, die Ihre Eingaben aus dem eben erstellten Formular in der Datenbank speichert. Diese PHP-Datei hat als einzige Aufgabe, den Inhalt des Hash, der vom Formular erzeugt wurde, in die Datenbank wieder zu speichern.

Legen Sie folgende Datei ebenfalls im `admin` Verzeichnis als `aendern-db.php` an:

```
<html>
<body>
<table>
<?php
$dbh=mysql_connect("localhost","pizza","passwort");
mysql_select_db("pizza");
foreach($Verfuegbar as $zutat=>$auswahl){
        mysql_query("UPDATE zutaten SET Verfuegbar='$auswahl'
                WHERE Zutat='$zutat'");
        echo "<tr><td>$zutat</td><td>$auswahl</td></tr>\n";
}
?>
</table>
<a href="index.html">Startseite der Administration</a>
</body>
</html>
```

Mit der `foreach`-Schleife durchläuft diese Routine den Hash `$Verfuegbar`. Dieser Hash hat als Schlüssel die Zutat und als Wert `j` oder `n` gespeichert, je nachdem, was der Benutzer auf dem Formular vorher eingestellt hatte. Für jede der Zutaten in `$Verfuegbar` wird jetzt das SQL-Kommando `UPDATE` verwendet, um den Wert des Hash in die Tabelle zu schreiben. Dazu geben Sie nach `UPDATE` an, in welcher Tabelle Sie Werte verändern möchten (hier `zutaten`). Im Anschluss daran wird mit `SET` angegeben, welches Tabellenfeld (`Verfuegbar`) welchen neuen Wert bekommen soll (`='$auswahl'`). Ohne die Angabe von `WHERE` würde das `UPDATE`-Kommando allerdings auf alle Elemente der Tabelle `zutaten` angewendet werden. Daher geben Sie hier noch an, dass der aktuelle `UPDATE`-Befehl für den Datensatz angewendet werden soll, der beim Feldnamen `Zutat` den Wert der Variablen `$zutat` enthält. Steht in Ihrem Hash also beispielsweise

`$Verfuegbar[Salami]="n"`,

so wird als SQL-Kommando ausgeführt:

```
UPDATE zutaten SET Verfuegbar='n' WHERE Zutat='Salami'
```

Weiterführende Informationen zum SQL-Befehl `UPDATE` bekommen Sie im Abschnitt 4.8.20.

Zusätzlich zum `UPDATE`-Kommando wird noch eine Bildschirmausgabe der Zutaten und deren Werte mit dem `echo`-Kommando erzeugt.

Zuletzt sollten Sie noch die Datei `index.html` im Verzeichnis `admin` bearbeiten, sodass der Link zum Löschen jetzt *Ändern* heisst und auf die Datei `aendern.php` verweist.

1.6 Beispiel im weiteren Verlauf

Als Beispielprojekt dient in den folgenden Kapiteln eine Kursverwaltung. Um bereits jetzt einen Überblick über die geplanten Funktionen zu bekommen, soll dieses Projekt hier erläutert werden.

Die Problemstellung lautet wie folgt:

Sie führen diverse Schulungen durch und bieten diese über das Internet an. Interessenten sollen sich über eine Weboberfläche über die angebotenen Kurse informieren können: Name des Kurses, Anfangs- und Enddatum des Kurses, maximale Teilnehmerzahl, Dozent, Schulungsort und – nicht zu vergessen – der Preis pro Person.

Neben der Darstellung für den Benutzer ist auch ein administrativer Bereich vorgesehen, in dem Sie selbst Kurse anlegen, verändern und löschen können. Diese Änderungen sollen ebenfalls über eine Weboberfläche durchgeführt und dann direkt in die Datenbank eingetragen werden. Dieser administrative Bereich soll natürlich nicht jedermann zur Verfügung stehen. Daher müssen für diesen Teil besondere Sicherheitsvorkehrungen getroffen werden.

Zu guter Letzt soll ein Benutzer auch in der Lage sein, einen dieser Kurse gleich über die Weboberfläche zu buchen. Die Eintragungen in dieser Buchungsmaske sollen ebenfalls wieder in einer Tabelle Ihrer Datenbank abgelegt werden. Dazu gehört zum einen, dass der Teilnehmer angelegt wird, und zum anderen, dass diesem Teilnehmer auch der entsprechende Kurs zugewiesen wird.