Schul Projekt

Projektübersicht

Für mein Schulprojekt habe ich eine fiktive Shopping-Seite erstellt, die es Nutzern ermöglicht, Produkte anzusehen, sich zu registrieren und Bestellungen aufzugeben. Die Webseite wurde mithilfe von HTML, CSS und PHP entwickelt und nutzt eine MySQL-Datenbank zur Verwaltung der Daten. Im Folgenden beschreibe ich die Implementierung der verschiedenen Komponenten der Webseite.

Projekt Struktur:

```
| _ Datenbank.sql

| _ dbConfig.php

| _ index.html

| _ home.php

| _ registrieren.php

| _ bestellen.php

| _ neues_produkt_einfügen.php

| _ README.md

| _ Dokumentation.md
```

```
| _ bilder

| _ lamp.jpg

| _ lock.jpg

| _ plug.png

| _ speaker.jpg

| _ thermostat.jpg
```

Datenbank.sql

Datenbankstruktur

- Ort: Speichert Postleitzahlen und Ortsnamen.
- Anschrift: Enthält Adressen mit Straße, Hausnummer und Verweis auf die Tabelle Ort.
- Kunde: Speichert Kundendaten inklusive Vorname, Nachname, E-Mail und Verweis auf die Anschrift.
- Artikel: Beinhaltet Produktinformationen wie Bezeichnung, Beschreibung, Preis und Bild.
- Bestellung: Verzeichnet Bestellungen mit Kunde und Datum.
- Position: Speichert die einzelnen Positionen einer Bestellung mit Verweis auf Artikel und Bestellung.

Die Datenbank für das Projekt enthält mehrere Tabellen zur Verwaltung von Kunden, Produkten, Bestellungen und

Adressen. Die Datenbank wird mit dem folgenden Skript erstellt:

Schritt 1: Erstellen der Datenbank

Zunächst habe ich sichergestellt, dass keine alte Version der Datenbank existiert, indem ich sie gegebenenfalls gelöscht habe. Anschließend habe ich eine neue Datenbank mit dem Namen "shop" erstellt und aktiviert.

```
DROP DATABASE IF EXISTS shop;

CREATE DATABASE shop;

USE shop;
```

Schritt 2: Erstellen der Tabelle "Ort"

Die erste Tabelle, die ich erstellt habe, ist die Tabelle "Ort". Diese enthält die Postleitzahl (PLZ) und den Namen des Ortes. Die PLZ und der Name dienen gemeinsam als Primärschlüssel, um sicherzustellen, dass jede Kombination einzigartig ist.

```
CREATE TABLE IF NOT EXISTS Ort (

PLZ int(11) NOT NULL,

Name varchar(100) NOT NULL,
```

```
PRIMARY KEY (PLZ, Name)

);
```

Schritt 3: Erstellen der Tabelle "Anschrift"

Als nächstes habe ich die Tabelle "Anschrift" erstellt, die eine eindeutige ID (AnsID), die Postleitzahl (PLZ), die Straße und die Hausnummer enthält. Die PLZ ist ein Fremdschlüssel, der auf die Tabelle "Ort" verweist.

```
CREATE TABLE IF NOT EXISTS Anschrift (

AnsID int(11) NOT NULL AUTO_INCREMENT,

PLZ int(11),

Straße varchar(100) NOT NULL,

Hausnummer int(11) NOT NULL,

PRIMARY KEY (AnsID),

FOREIGN KEY (PLZ) REFERENCES Ort(PLZ)

);
```

Schritt 4: Erstellen der Tabelle "Kunde"

Die Tabelle "Kunde" enthält die Kundennummer (KNr), den Vor- und Nachnamen, die Anschriften-ID (AnsID) und die E-Mail-Adresse. Die AnsID ist ein Fremdschlüssel, der auf die Tabelle "Anschrift" verweist.

```
CREATE TABLE IF NOT EXISTS Kunde (

KNr int(11) NOT NULL AUTO_INCREMENT,

Vorname varchar(100) NOT NULL,

Nachname varchar(100) NOT NULL,

AnsID int(11),

Email varchar(100) NOT NULL,

PRIMARY KEY (KNr),

FOREIGN KEY (AnsID) REFERENCES Anschrift(AnsID)

);
```

Schritt 5: Erstellen der Tabelle "Artikel"

Die Tabelle "Artikel" enthält die Artikelnummer (ANr), die Bezeichnung, eine Beschreibung, den Preis und einen Pfad zum Bild. Diese Tabelle dient zur Speicherung der Produkte, die im Online-Shop angeboten werden.

```
CREATE TABLE IF NOT EXISTS Artikel (

ANr int(11) NOT NULL AUTO_INCREMENT,

Bezeichnung varchar(100) NOT NULL,

Beschreibung varchar(1000),

Preis double(10,2) NOT NULL,

Bild varchar(255),

PRIMARY KEY (ANr)

);
```

Schritt 6: Erstellen der Tabelle "Bestellung"

In der Tabelle "Bestellung" werden die Bestellnummer (BesNr), die Kundennummer (KNr) und das Datum der Bestellung gespeichert. Die KNr ist ein Fremdschlüssel, der auf die Tabelle "Kunde" verweist.

```
CREATE TABLE IF NOT EXISTS Bestellung (

BesNr int(11) NOT NULL AUTO_INCREMENT,

KNr int(11),

Datum datetime NOT NULL,
```

```
PRIMARY KEY (BesNr),

FOREIGN KEY (KNr) REFERENCES Kunde(KNr)

);
```

Schritt 7: Erstellen der Tabelle "Position"

Die letzte Tabelle, die ich erstellt habe, ist die Tabelle "Position". Diese enthält die Bestellpositionsnummer (BesPos), die Artikelnummer (ANr), die Bestellnummer (BesNr) und die Anzahl der Artikel. Die ANr und BesNr sind Fremdschlüssel, die auf die Tabellen "Artikel" bzw. "Bestellung" verweisen.

```
CREATE TABLE IF NOT EXISTS Position (

BesPos int(11) NOT NULL AUTO_INCREMENT,

ANr int(11),

BesNr int(11),

Anzahl int(11) NOT NULL,

PRIMARY KEY (BesPos),

FOREIGN KEY (ANr) REFERENCES Artikel(ANr),

FOREIGN KEY (BesNr) REFERENCES Bestellung(BesNr)
```

```
);
```

Schritt 8: Einfügen von Testdaten

Um die Funktionsweise der Datenbank zu überprüfen, habe ich einige Testdaten in die Tabelle "Artikel" und die Tabelle "Kunde" eingefügt.

```
INSERT INTO Artikel (ANr, Bezeichnung,
Beschreibung, Preis, Bild)

VALUES

(001, "Smart Speaker", "Der Echo Studio
kombiniert High-Fidelity-Klang mit den
Funktionen von verschiedenen Sprachsteuerungen.
Mit seinem 3D-Klangsystem füllt er den Raum mit
beeindruckendem Sound. Dank der Sprachsteuerung
können Benutzer per Sprachbefehl Musik
abspielen, Fragen stellen und Smart-Home-Geräte
steuern. Der Echo Studio passt sich automatisch
an die Raumakustik an und bietet so stets
optimalen Klang. Ideal für Audiophile und Smart-
Home-Enthusiasten.", 49.99,
"../bilder/speaker.jpg"),

(002, "Smart Plug", "Der Mini Smart Plug
ermöglicht die Fernsteuerung angeschlossener
Geräte über das Internet. Mit der zugehörigen
App können Sie Geräte ein- und ausschalten,
```

```
Zeitpläne erstellen und den Energieverbrauch
überwachen.",33.13, "../bilder/plug.png"),

(003, "Smart Lock", "Der Wi-Fi Smart Lock
ermöglicht die sichere Steuerung Ihrer Tür von
überall aus über das Internet. Mit der
zugehörigen App können Sie die Tür verriegeln
und entriegeln, virtuelle Schlüssel verwalten
und Aktivitätsprotokolle überprüfen.", 159.99,
"../bilder/lock.jpg"),

(004, "Smart Lamp", "Die Smart Lamp bietet eine
einfache Möglichkeit, Ihre Beleuchtung zu
automatisieren und zu steuern. Über die Philips
Hue-App können Sie die Lampe ein- und
ausschalten, Helligkeit und Farbtemperatur
anpassen sowie Zeitpläne festlegen.", 14.99,
"../bilder/lamp.jpg"),

(005, "Smart Thermostat", "Das Smart Thermostat
ermöglicht die intelligente Steuerung Ihrer
Heizung und Kühlung von überall aus über das
Internet. Mit der zugehörigen App können Sie die
Temperatur einstellen, Zeitpläne programmieren
und Energieverbrauchsberichte anzeigen.", 39.99,
"../bilder/thermostat.jpg");



INSERT INTO Kunde (KNr, Vorname, Nachname,
Email)

VALUES (0, "admin", "admin", "admin@gmail.com");
```

Diese Schritte dokumentieren, wie ich die Datenbank für den Online-Shop erstellt habe. Jede Tabelle wurde sorgfältig geplant und erstellt, um eine konsistente und effiziente Datenverwaltung zu gewährleisten. Die Verwendung von Fremdschlüsseln stellt sicher, dass die Datenintegrität über die verschiedenen Tabellen hinweg gewahrt bleibt.

Datenbankkonfiguration

Die Verbindung zur Datenbank wird in einer separaten Konfigurationsdatei eingerichtet. Diese Datei enthält die Zugangsdaten und stellt sicher, dass die Webseite auf die Datenbank zugreifen kann.

dbConfig.php:

Eine Konfigurationsdatei (`dbConfig.php`) wurde erstellt, um die Verbindung zur MySQL-Datenbank für die "Smart GmbH" Website zu verwalten. Diese Datei enthält wichtige Informationen zur Datenbankverbindung und stellt sicher, dass die Website mit der Datenbank kommunizieren kann.

PHP-Tag und Datenbankdetails

Die Datei beginnt mit dem Öffnen des PHP-Tags. Danach werden die Details der Datenbankverbindung definiert, einschließlich des Hostnamens, des Benutzernamens, des Passworts und des Datenbanknamens. Da es sich um eine lokale Entwicklung handelt, bleibt das Passwortfeld leer.

```
<?php
```

```
//DB details

$dbHost = 'localhost'; // Link

$dbUsername = 'root'; // Link

$dbPassword = ''; // Ist Leer weil es im
LocalHost ist

$dbName = 'shop'; // Name der Datenbank
```

Erstellen der Datenbankverbindung

Im nächsten Schritt erstelle ich eine neue Verbindung zur MySQL-Datenbank mithilfe der `mysqli` Klasse und den zuvor definierten Verbindungsdetails.

```
// Erschaffe eine Verbindung zur Datenbank

$db = new mysqli($dbHost, $dbUsername,
$dbPassword, $dbName);
```

Fehlerbehandlung

Falls die Verbindung zur Datenbank fehlschlägt, gebe ich eine Fehlermeldung aus und beende das Skript. Dies wird durch die `connect_error` Eigenschaft der `mysqli` Klasse ermöglicht.

```
// Error Nachricht wenn keine verbindung mit der Datenbank möglich war

if ($db->connect_error) {

die('Verbindung fehlgeschlagen: ' . $db->connect_error);

}

?>
```

index.html

Dies ist die Startseite der Webseite. Sie enthält eine Navigation und eine Willkommensnachricht.

```
<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Smart GmbH</title>

<link rel="stylesheet" href="style.css">

</head>
```

```
<body>

<div class="form">

<nav class="navbar navbar-inverse"
style="border-radius: 0px;">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" class="active"
href="index.html">Smart GmbH</a>

</div>

<ul class="nav navbar-nav">

<li>

<a href="home.php">Home (Produkte)</a>

</li>

<li>

<a href="registrieren.php">Registrierung</a>

</li>

<li>

<a href="bestellen.php">Bestellformular</a>
```

```
</li>

</ul>

</div>

</nav>

<h1 style="margin-top: 30vh;">Wilkommen</h1>

<br>

<p style="margin-bottom: 50vh;">Lorem ipsum
dolor sit amet consectetur adipisicing elit.
Voluptates cupiditate ab eaque! Praesentium
ducimus exercitationem non ratione, doloremque
optio ipsa porro aut provident ut dolore fuga
dolorem aliquid amet modi. Assumenda, illo.
Blanditiis iusto excepturi deleniti minima.
Obcaecati dicta recusandae commodi quos rerum
hic, magnam ipsum adipisci quae ut voluptatum
tempore culpa reiciendis. Quo commodi blanditiis
assumenda animi facere perferendis laboriosam
sit sequi quaerat porro. Corporis assumenda,
ullam illo veniam facere, ratione nihil animi
quisquam nisi ut autem soluta blanditiis dolores
deserunt. In sit illo ullam dolor magnam sint
sequi voluptatem, quos beatae aperiam iusto.
Sequi optio tempora rem non!</p>

</div>

</body>
```

```
</html>
```

home.php

Diese Datei zeigt die verfügbaren Produkte an. Sie ruft die Produktdaten aus der Datenbank ab und zeigt sie in einer strukturierten Liste an.

PHP-Abschnitt: Einbinden der Datenbankkonfigurationsdatei

Zu Beginn des PHP-Codes binde ich die Datei `dbConfig.php` ein, die die Konfigurationsdetails für die Datenbankverbindung enthält. Dies ermöglicht es, die Datenbankverbindung in dieser Datei zu nutzen.

```
<?php

include "dbConfig.php"; // Einbinden der
Datenbankkonfigurationsdatei

?>
```

HTML-Abschnitt: Grundstruktur

Der HTML-Teil der Datei beginnt mit dem `<!DOCTYPE html>` Tag, um den Browser anzuweisen, das Dokument als HTML5 zu interpretieren.

```
<!DOCTYPE html>

<html lang="de">
```

HTML-Abschnitt: Kopfbereich (`<head>`)

Im Kopfbereich der Seite setze ich den Zeichensatz auf UTF-8 und definiere den Titel der Seite. Außerdem binde ich eine CSS-Datei (`style.css`) ein, um das Styling der Seite zu ermöglichen. Das `meta`-Tag `viewport` sorgt dafür, dass die Seite auf verschiedenen Geräten gut aussieht.

```
<head>

<meta charset="utf8">

<title>Smarthome Produkte - Smart GmBH</title>

<meta name="viewport" content="width=device-
width, initial-scale=1">

<link rel="stylesheet" href="style.css">

</head>
```

HTML-Abschnitt: Körperbereich (`<body>`)

Der Hauptteil der Seite beginnt mit dem `<body>` Tag.

```
<body>
```

Navigationsleiste (`<nav>`)

Die Navigationsleiste ermöglicht es den Benutzern, zwischen verschiedenen Seiten der Website zu wechseln. Sie enthält Links zur Startseite (`index.html`), zur Registrierung (`registrieren.php`) und zum Bestellformular (`bestellen.php`).

```
<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="index.html">Smart GmbH</a>

</div>



<ul class="nav navbar-nav">

<li class="active">

<a href="home.php">Home (Produkte)</a>

</li>
```

```
<li>

<a href="registrieren.php">Registrierung</a>

</li>

<li>

<a href="bestellen.php">Bestellformular</a>

</li>

</ul>

</div>

</nav>
```

Hauptinhalt: Produkte anzeigen

In diesem Abschnitt werden die Produkte aus der Datenbank abgerufen und angezeigt.

```
<div class="container">

<h1>Unsere Produkte</h1>

<br>
```

```
<div id="products" class="row list-group">
```

PHP-Abschnitt: Datenbankabfrage und Produktanzeige

Hier wird eine SQL-Abfrage ausgeführt, um die Produkte aus der Tabelle `Artikel` abzurufen. Die Ergebnisse der Abfrage werden in einer Schleife durchlaufen und angezeigt.

```
<?php

$query = $db->query("SELECT * FROM Artikel ORDER
BY ANr LIMIT 10");



if ($query->num_rows > 0) {

while ($row = $query->fetch_assoc()) { ?>
```

Für jedes Produkt wird ein HTML-Block erstellt, der das Bild, die Bezeichnung, die Beschreibung, die Produkt-ID und den Preis enthält.

```
<div class="item col-lg-4">

<div class="thumbnail">
```

```
<img src="<?php echo $row['Bild']; ?>" alt="<?
php echo htmlspecialchars($row['Bild']); ?>"
class="product-image">

<div class="caption">

<h4 class="list-group-item-heading"><?php echo
htmlspecialchars($row['Bezeichnung']); ?></h4>

<p class="list-group-item-text" style="padding-
bottom:10px"><?php echo
htmlspecialchars($row['Beschreibung']); ?></p>

<h3 class="list-group-item-heading">Produkt ID:
<?php echo htmlspecialchars($row['ANr']); ?>
</h3>

<div class="row">

<div class="col-md-6">

<p class="lead"><?php echo $row['Preis'] . ' €';
?></p>

</div>

</div>

</div>

</div>

</div>
```

```
<?php }

} else { ?>

<p>Produkte wurden nicht gefunden.</p>

<?php }

?>
```

Abschließende Tags und Footer

Zum Schluss werden die `div`, `body` und `html` Tags von allen html und php Dateien geschlossen. Der Footer enthält Informationen über wer alles, was aus unserer Gruppe gemacht hat.

```
</div>

<!-- Footer Sektion -->

<footer>

<p>Developed by Gökdeniz and Ralf. Databank
modelling by Adrian and Elias, Project
Management by Natalie</p>

</footer>
```

```
</body>

</html>
```

registrieren.php

registrieren.php` ermöglicht es Nutzern, sich auf unsere Website zu registrieren. Diese Seite enthält ein Formular zur Eingabe von Benutzerdaten und verarbeitet die Registrierungsinformationen, indem sie diese in eine Datenbank speichert. Hier ist eine Schritt-für-Schritt-Erklärung des Codes.

PHP-Abschnitt: Einbinden der Datenbankkonfigurationsdatei

Zuerst binde ich die Datei `dbConfig.php` ein, die die Konfigurationsdetails für die Datenbankverbindung enthält. Dies ermöglicht es, die Datenbankverbindung in dieser Datei zu nutzen.

```
<?php

include "dbConfig.php"; //
Datenbankkonfiguration einbinden

?>
```

Überprüfen, ob das Formular abgeschickt wurde

Ich überprüfe, ob das Formular mit der POST-Methode abgeschickt wurde. Wenn dies der Fall ist, speichere ich die POST-Daten in Variablen.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {

$vorname = $_POST["vorname"];

$nachname = $_POST["nachname"];

$email = $_POST["email"];

$strasse = $_POST["strasse"];

$hausnummer = $_POST["hausnummer"];

$plz = $_POST["plz"];

$ort = $_POST["ort"];
```

Überprüfen, ob der Benutzername oder die E-Mail bereits existieren

Ich führe eine SQL-Abfrage aus, um zu überprüfen, ob der Vorname oder die E-Mail-Adresse bereits in der Tabelle `Kunde` existiert.

```
$sqlCheckUser = "SELECT * FROM Kunde WHERE
Vorname = ? OR Email = ?";
```

```
$stmtCheckUser = $db->prepare($sqlCheckUser);

$stmtCheckUser->bind_param("ss", $vorname,
$email);

$stmtCheckUser->execute();

$resultCheckUser = $stmtCheckUser->get_result();



if ($resultCheckUser->num_rows > 0) {

$errorMessage = "Der Benutzername oder die E-
Mail-Adresse sind bereits vergeben. Bitte wählen
Sie einen anderen Namen oder eine andere E-
Mail.";

} else {

$db->autocommit(FALSE);



try {
```

Überprüfen, ob die PLZ bereits existiert

Ich überprüfe, ob die eingegebene PLZ bereits in der Tabelle `Ort` existiert. Wenn nicht, füge ich einen neuen Eintrag für die PLZ und den Ort hinzu.

```
$sqlCheckPLZ = "SELECT PLZ FROM Ort WHERE PLZ =
?";

$stmtCheckPLZ = $db->prepare($sqlCheckPLZ);

$stmtCheckPLZ->bind_param("i", $plz);

$stmtCheckPLZ->execute();

$resultCheckPLZ = $stmtCheckPLZ->get_result();



if ($resultCheckPLZ->num_rows === 0) {

$sqlInsertPLZ = "INSERT INTO Ort (PLZ, Name)
VALUES (?, ?)";

$stmtInsertPLZ = $db->prepare($sqlInsertPLZ);

$stmtInsertPLZ->bind_param("is", $plz, $ort);

$stmtInsertPLZ->execute();

}
```

Einfügen der Anschrift und Kundendaten in die Datenbank

Ich füge die Anschrift in die Tabelle `Anschrift` ein und speichere die generierte ID. Anschließend füge ich die

Kundendaten in die Tabelle `Kunde` ein und speichere die Kundennummer.

```
$sqlAnschrift = "INSERT INTO Anschrift (PLZ,
Straße, Hausnummer) VALUES (?, ?, ?)";

$stmtAnschrift = $db->prepare($sqlAnschrift);

$stmtAnschrift->bind_param("iss", $plz,
$strasse, $hausnummer);

$stmtAnschrift->execute();

$anschriftId = $db->insert_id;



$sqlKunde = "INSERT INTO Kunde (Vorname,
Nachname, AnsID, Email) VALUES (?, ?, ?, ?)";

$stmtKunde = $db->prepare($sqlKunde);

$stmtKunde->bind_param("ssis", $vorname,
$nachname, $anschriftId, $email);

$stmtKunde->execute();

$kundennummer = $db->insert_id;



$db->commit();
```

```
$successMessage = "Registrierung erfolgreich! Ihre Kundennummer lautet: $kundennummer. Bitte notieren Sie sich diese ID-Nummer.";

} catch (Exception $e) {

$db->rollback();

$errorMessage = "Fehler beim Registrieren: " . $e->getMessage();

}

}

}

?>
```

HTML-Abschnitt: Grundstruktur

Der HTML-Teil der Datei beginnt mit dem `<!DOCTYPE html>` Tag und setzt die Sprache auf Deutsch.

```
<!DOCTYPE html>

<html lang="de">
```

HTML-Abschnitt: Kopfbereich (`<head>`)

Im Kopfbereich setze ich den Zeichensatz auf UTF-8, definiere den Titel der Seite und binde eine CSS-Datei (`style.css`) ein.

```
<head>
<meta charset="UTF-8">
<title>Registrieren</title>
<link rel="stylesheet" href="style.css">
</head>
```

HTML-Abschnitt: Körperbereich (`<body>`)

Der Hauptteil der Seite beginnt mit dem `<body>` Tag.

```
<body>
```

Navigationsleiste (`<nav>`)

Die Navigationsleiste ermöglicht es den Benutzern, zwischen verschiedenen Seiten der Website zu wechseln.

```
<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">

<a class="navbar-brand" href="index.html">Smart
GmbH</a>

</div>



<ul class="nav navbar-nav">

<li>

<a href="home.php">Home (Produkte)</a>

</li>

<li class="active">

<a href="registrieren.php">Registrierung</a>

</li>

<li>

<a href="bestellen.php">Bestellformular</a>

</li>
```

```
</ul>

</div>

</nav>
```

Formular zur Kundenregistrierung

Das Formular ermöglicht es den Benutzern, ihre Registrierungsdaten einzugeben. Es enthält Felder für Vorname, Nachname, E-Mail, Straße, Hausnummer, PLZ und Ort.

```
<h1>Kundenregistrierung</h1>



<form method="POST">

<label for="vorname">Vorname:</label>

<input type="text" id="vorname" name="vorname"
required><br>



<label for="nachname">Nachname:</label>

<input type="text" id="nachname" name="nachname"
required><br>
```

```
<label for="email">Email:</label>

<input type="email" id="email" name="email"
required><br>



<h2>Anschrift</h2>

<label for="strasse">Straße:</label>

<input type="text" id="strasse" name="strasse"
required><br>



<label for="hausnummer">Hausnummer:</label>

<input type="number" id="hausnummer"
name="hausnummer" required><br>



<label for="plz">PLZ:</label>

<input type="number" id="plz" name="plz"
required><br>



<label for="ort">Ort:</label>
```

```
<input type="text" id="ort" name="ort" required>
<br>



<input type="submit" value="Registrieren">

</form>
```

Erfolgs- oder Fehlermeldungen anzeigen

Wenn die Registrierung erfolgreich war oder ein Fehler aufgetreten ist, wird eine entsprechende Nachricht angezeigt und die Seite neu geladen.

```
<?php if (isset($successMessage) ||
isset($errorMessage)): ?>

<script>

alert('<?php echo $successMessage ??
$errorMessage; ?>');

window.location.reload();

</script>

<?php endif; ?>
```

Footer

Der Footer enthält Informationen über die Entwickler des Projekts.

```
<footer>

<p>Developed by Gökdeniz and Ralf. Databank modelling by Adrian and Elias, Project Management by Natalie</p>

</footer>

</body>

</html>
```

bestellen.php

Produkte bestellen. Diese Seite nutzt eine Datenbankverbindung, um Bestellungen zu speichern, und verwendet Sessions, um den Warenkorb zu verwalten. Hier ist eine detaillierte Erklärung des Codes.

PHP-Abschnitt: Session starten und Datenbankkonfigurationsdatei einbinden

Zuerst starte ich die Session, um Sitzungsdaten zu speichern, und binde die Datei `dbConfig.php` ein, die die Konfigurationsdetails für die Datenbankverbindung enthält.

```php
<?php

session_start(); // Starten der Session, um Sitzungsdaten zu speichern

include "dbConfig.php"; // Einbinden der Datenbankkonfigurationsdatei

?>
```

Initialisieren des Warenkorbs

Ich überprüfe, ob der Warenkorb bereits in der Session existiert. Falls nicht, initialisiere ich einen leeren Warenkorb und setze die Gesamtsumme auf 0.

```php
// Erstelle zuerst ein leeren Warenkorp

if (!isset($_SESSION['warenkorb'])) {

$_SESSION['warenkorb'] = array();

$_SESSION['gesamtsumme'] = 0;

}



$productNotFound = false; // Flag für nicht
```

```
gefundenes Produkt
```

Überprüfen, ob das Formular abgeschickt wurde

Ich überprüfe, ob das Formular mit der POST-Methode abgeschickt wurde. Es gibt drei mögliche Aktionen: Speichern eines Produkts im Warenkorb, Entfernen eines Produkts aus dem Warenkorb und Abschließen einer Bestellung.

Speichern eines Produkts im Warenkorb

Wenn das Produktformular abgeschickt wurde, hole ich die Produktinformationen aus der Datenbank und speichere das Produkt im Warenkorb.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {

if (isset($_POST['speichern'])) {

$produktId = $_POST["produktId"];

$menge = $_POST["menge"];

// Artikelinformationen aus der Datenbank
abrufen

$sql = "SELECT Bezeichnung, Preis FROM Artikel
WHERE ANr = ?";

$stmt = $db->prepare($sql);
```

```
$stmt->bind_param("i", $produktId);

$stmt->execute();

$result = $stmt->get_result();

if ($result->num_rows > 0) {

$row = $result->fetch_assoc();

$gesamtPreis = $row['Preis'] * $menge;

// Artikel und Menge in der Session speichern

$_SESSION['warenkorb'][] = array('produktId' =>
$produktId, 'Bezeichnung' =>
$row['Bezeichnung'], 'menge' => $menge, 'gesamt'
=> $gesamtPreis);

$_SESSION['gesamtsumme'] += $gesamtPreis;

} else {

$productNotFound = true;

}
```

Entfernen eines Produkts aus dem Warenkorb

Wenn das Entfernen-Formular abgeschickt wurde, entferne ich das Produkt aus dem Warenkorb und aktualisiere die Gesamtsumme.

```
} elseif (isset($_POST['entfernen'])) {

// Produkt aus dem Warenkorb entfernen

$indexToRemove = $_POST['index'];

if (isset($_SESSION['warenkorb']
[$indexToRemove])) {

$_SESSION['gesamtsumme'] -=
$_SESSION['warenkorb'][$indexToRemove]
['gesamt'];

array_splice($_SESSION['warenkorb'],
$indexToRemove, 1);

}
```

Abschließen einer Bestellung

Wenn das Bestellformular abgeschickt wurde, speichere ich die Bestellung und die einzelnen Positionen in der Datenbank.

```
} elseif (isset($_POST['bestellen']) &&
!empty($_SESSION['warenkorb'])) {

$kundennummer = $_POST["kundennummer"]; //
Kundennummer aus dem separaten Formular
```

```php
// Bestellung in die Datenbank einfügen

$db->begin_transaction();

try {

$sqlBestellung = "INSERT INTO Bestellung (KNr,
Datum) VALUES (?, NOW())";

$stmtBestellung = $db->prepare($sqlBestellung);

$stmtBestellung->bind_param("i", $kundennummer);

$stmtBestellung->execute();

$bestellId = $stmtBestellung->insert_id;



foreach ($_SESSION['warenkorb'] as $item) {

$sqlPosition = "INSERT INTO Position (ANr,
BesNr, Anzahl) VALUES (?, ?, ?)";

$stmtPosition = $db->prepare($sqlPosition);

$stmtPosition->bind_param("iii",
$item['produktId'], $bestellId, $item['menge']);

$stmtPosition->execute();

}
```

```
$db->commit();

displayWarenkorb(); // Zeige den Warenkorb mit
der Bestellung an

$_SESSION['warenkorb'] = array();

$_SESSION['gesamtsumme'] = 0;

echo "<script>alert('Bestellung erfolgreich
aufgegeben!'); window.location.reload();
</script>";

} catch (Exception $e) {

$db->rollback();

echo "<script>alert('Fehler beim Bestellen: " .
$e->getMessage() . "');</script>";

}

}

}
```

Funktion zur Anzeige des Warenkorbs

Ich habe eine Funktion `displayWarenkorb` erstellt, um den aktuellen Inhalt des Warenkorbs anzuzeigen.

```
function displayWarenkorb() {

if (!empty($_SESSION['warenkorb'])) {

echo "<h3>Warenkorb:</h3>";

echo "<ul>";

foreach ($_SESSION['warenkorb'] as $index =>
$item) {

echo "<li>Produkt ID: " . $item['produktId'] .
", Produktname: " . $item['Bezeichnung'] . ",
Menge: " . $item['menge'] . ", Gesamt: " .
$item['gesamt'] . "€ ";

echo "<form method='POST' style='display:
inline;'><input type='hidden' name='index'
value='$index'><input type='submit'
name='entfernen' value='Entfernen'></form>
</li>";

}

echo "</ul>";

echo "<h3>Gesamtsumme: " .
$_SESSION['gesamtsumme'] . "€</h3>";

}

}
```

```
?>
```

HTML-Abschnitt: Grundstruktur

Der HTML-Teil der Datei beginnt mit dem `<!DOCTYPE html>` Tag und setzt die Sprache auf Deutsch.

```
<!DOCTYPE html>

<html lang="de">
```

HTML-Abschnitt: Kopfbereich (`<head>`)

Im Kopfbereich setze ich den Zeichensatz auf UTF-8, definiere den Titel der Seite und binde eine CSS-Datei (`style.css`) ein.

```
<head>

<meta charset="UTF-8">

<title>Produkt bestellen</title>

<script>

window.onload = function() {

<?php if ($productNotFound) { ?>
```

```
alert("Produkt nicht gefunden.");

<?php } ?>

};

</script>

<link rel="stylesheet" href="style.css">

</head>
```

HTML-Abschnitt: Körperbereich (`<body>`)

Der Hauptteil der Seite beginnt mit dem `<body>` Tag.

```
<body>
```

Navigationsleiste (`<nav>`)

Die Navigationsleiste ermöglicht es den Benutzern, zwischen verschiedenen Seiten der Website zu wechseln.

```
<nav class="navbar navbar-inverse">

<div class="container-fluid">

<div class="navbar-header">
```

```
<a class="navbar-brand" href="index.html">Smart
GmbH</a>

</div>



<ul class="nav navbar-nav">

<li>

<a href="home.php">Home (Produkte)</a>

</li>

<li>

<a href="registrieren.php">Registrierung</a>

</li>

<li class="active">

<a href="bestellen.php">Bestellformular</a>

</li>

</ul>

</div>

</nav>
```

Formular zur Produktauswahl

Dieses Formular ermöglicht es den Benutzern, die Produkt-ID und die Menge einzugeben, um ein Produkt in den Warenkorb zu legen.

```
<h1>Produkt bestellen</h1>



<form method="POST">

<label for="produktId">Produkt ID:</label>

<input type="number" id="produktId"
name="produktId" required><br>



<label for="menge">Menge:</label>

<input type="number" id="menge" name="menge"
required><br>



<input type="submit" name="speichern"
value="Speichern">

</form>
```

Formular zur Eingabe der Kundennummer

Dieses Formular ermöglicht es den Benutzern, ihre Kundennummer einzugeben und die Bestellung abzuschließen.

```
<form method="POST">

<label for="kundennummer">Kundennummer:</label>

<input type="number" id="kundennummer"
name="kundennummer" required><br>



<input type="submit" name="bestellen"
value="Bestellen" onclick="return confirm('Sind
Sie sicher, dass Sie die Bestellung aufgeben
möchten?');">

</form>
```

Warenkorb anzeigen

Hier wird der aktuelle Inhalt des Warenkorbs angezeigt.

```
<?php displayWarenkorb(); ?>
```

style.css

Hier is das Styling der Website der "Smart GmbH" definiert. Diese Datei enthält verschiedene CSS-Regeln, um das Layout und das Aussehen der HTML-Elemente zu gestalten. Hier ist eine detaillierte Erklärung des Codes.

Grundlegende Farbdefinitionen und Medienabfragen

Zuerst definiere ich einige CSS-Variablen, um die Hintergrundfarbe und Textfarbe der Seite zu speichern. Diese Variablen ändern sich je nach dem bevorzugten Farbschema (hell oder dunkel) des Benutzers.

```
:root {
--background-color: #F8F8F8;
--text-color: #0f0f0f;
}

@media (prefers-color-scheme: dark) {
:root {
--background-color: #0f0f0f;
--text-color: #F8F8F8;
}
```

```
}
```

Allgemeine Stile

Hier setze ich einige allgemeine Stile für alle HTML-Elemente, um ein einheitliches Erscheinungsbild zu gewährleisten.

```
* {
box-sizing: border-box;
}

html {
overflow-y: scroll;
overflow-x: hidden;
}

body {
background-color: var(--background-color);
color: var(--text-color);
```

```
max-width: 100vw;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

flex-direction: column;

margin: 0;

}
```

Stile für Links

Ich definiere das Aussehen von Links und füge Übergangseffekte hinzu, um das Benutzererlebnis zu verbessern.

```
a {

text-decoration: none;

color: #b11a1a;

-webkit-transition: .5s ease;
```

```
transition: .5s ease;

}
```

Stile für Überschriften und Absätze

Hier definiere ich die Stile für verschiedene Überschriften und Absätze, um die Textdarstellung auf der Seite zu verbessern.

```
h1 {

text-align: center;

font-weight: 300;

margin: 0 0 40px;

}



h2 {

text-align: center;

font-weight: 1000;

margin: 0;

}
```

```
p {
text-align: center;
margin: 0px 0px 50px 0px;
}
```

Stile für die Navigationsleiste

Ich gestalte die Navigationsleiste, um sie benutzerfreundlich und optisch ansprechend zu machen.

```
.navbar {
width: 100%;
border-bottom: var(--text-color) 1px solid;
margin-bottom: 10vh;
}

.container-fluid {
display: flex;
```

```
justify-content: space-between;

align-items: center;

padding-left: 8%;

padding-right: 8%;

}



.navbar-header {

flex-grow: 1;

}



.navbar-nav {

list-style: none;

display: flex;

padding: 0;

margin: 0;

}
```

```
.navbar-nav li {

padding: 10px;

}



.navbar-nav li a, .navbar-header a {

text-decoration: none;

}



.navbar-nav li a:hover, .navbar-header a:hover {

-webkit-filter: brightness(50%);

filter: brightness(50%);

}



.navbar-nav li.active a {

color: var(--text-color);

}
```

Container-Stile

Der Container-Stil sorgt dafür, dass der Inhalt der Seite zentriert und gut lesbar ist.

```
.container {

text-align: center;

padding-left: 10vw;

padding-right: 10vw;

max-width: 90vw;

margin: auto;

min-width: 70vw;

margin-left: 50%;

transform: translateX(-50%);

border-radius: 20px;

}
```

Stile für Thumbnails und Produktbilder

Hier definiere ich die Stile für Thumbnails und Produktbilder, um sicherzustellen, dass sie einheitlich und ansprechend

aussehen.

```
.thumbnail {
border-radius: 20px;
padding: 20px;
margin-bottom: 20%;
}

.product-image {
width: 80%;
height: 20vw;
border-radius: 10px;
margin-top: -6%;
margin-bottom: 15px;
object-fit: cover;
transition: all 0.2s ease-in;
}
```

```
.product-image:hover {

scale: 1.008;

}
```

Stile für das Layout der Seite

Diese Regeln sorgen dafür, dass das Layout der Seite ansprechend und benutzerfreundlich ist.

```
.row {

justify-content: center;

margin-top: 10vh;

}



.item {

margin-top: 20px;

}



.caption {
```

```
text-align: center;

}
```

Stile für Formulare

Ich definiere die Stile für Formulare, um sicherzustellen, dass sie gut aussehen und einfach zu bedienen sind.

```
form {

border-radius: 10px;

margin: 10px;

width: 90%;

padding: 20px;

max-height: 100%;

height: 98%;

}



input, label {

display: block;
```

```
width: 100%;

margin-top: 10px;

padding: 1%;

}
```